PVAX2/PMariah Memory/Graphics/Video Controller

VAX Workstation Engineering

13–September–1990

This document is the hardware specification of the PVAX2/PMariah memory, graphics, and video controller.

Contacts:

Kim Meinerth,	Jacob::Meinerth,	ML05-2,	223-3328
Blaise Fanning,	Jacob::Fanning,	ML05-2,	223-2036
K. Srinivasan,	Jacob::Srini,	ML05-2,	223-2322
Ali Moezzi,	Jacob::Moezzi,	ML05-2,	223-2630
Colyn Case,	Star::Case,	ZK03-4,	381-1245
John Irwin,	Star::Irwin,	ZK03-4,	381-2813
Jonathan Kaplan,	Star::Kaplan,	ZK03-4,	381-1950
Chris Franklin,	Star::Franklin,	ZK03-4,	381-1952
Desty Masucci,	Leddev::Masucci,	ML021-4,	223-2824

digital equipment corporation, maynard, massachusetts

Preface

This document is the formal specification of the PVAX2/PMariah memory, graphics, and video controller hardware.

Document Structure

This document is arranged in sections, each of which describes a major topic of the PVAX2/PMariah memory, graphics, and video controller hardware.

Associated Documents

- **PVAX2 System Specification**—John Kirk
- **PMariah System Specification**—John Kirk
- **PVAX2/PMariah S-Chip Specification**—George Lord
- **PVAX2/PMariah LCG Address Genarator Design Specification**—Blaise Fanning
- **PVAX2/PMariah LCG Virtual Memory Support**—Colyn Case
- **PVAX2/PMariah LCG Command Packets**—Colyn Case
- **PVAX2/PMariah LCG HW Overview** (\ included here)—Kim Meinerth
- **PVAX2/PMariah LCG Pins, Interface & Internal Signals** (included here)—Kim Meinerth

Revision Information

Date	Editor	Description/Summary of Changes
20-AUG-1990	Kim Meinerth	updates
12-JAN-1989	John Irwin	Rework register section.
14-NOV-1989	Kim A. Meinerth & Jonathan C. Kaplan	Restructure GRAPHICS Registers, add MISC_ENABLE and MISC_STATUS, and merge in VMS register names
21-SEP-1989	John Irwin	Coverted to VAX-DOCUMENT format
08-SEP-1989	Kim A. Meinerth	Start PMariah Updates
10-JAN-1989	Kim A. Meinerth	Start SOC Updates
17-AUG-1988	Kim A. Meinerth	Initial Working Draft

1 Introduction

The Low Cost Graphics (LCG) system is jointly specified and architected with VMS DECwindows Engineering.

The PVAX2/PMariah Graphics (1, 8, 24 plane), Memory, and Video Controller is a portion of the PVAX2/PMariah System ASIC, S-Chip (DC7201). LCG is a smart memory controller which understands the most heavily used "X-Window" graphics primitives and executes them at memory bandwidth speeds with minimal memory latency. LCG provides competitive 2D Graphics performance and functionality for the lowest cost possible. The LCG architecture allows the graphics primitives supported to be greatly expanded in future implementations.

LCG operates on both frame buffer memory and virtual main memory. Graphics primitives are supported to multiple clipping rectangles (in frame buffer and virtual memory) for direct support of over-lapping windows and backing store with linear addressing. Main memory has byte parity for both lowest cost and graphics performance which tends to be byte (8 plane) oriented.

The memory system supports byte parity memory with up to 4 banks of 64 bit main memory, as well as, up to 4 frame buffers. This gives main memory configurations of 8M, 16M, 24M, and 32M bytes with 1Mx1 DRAMs which is the maximum memory range for PVAX2. The memory increments MUST be a multiple of 64 bits or 8M bytes.

The PMariah system can access up to 104M bytes with 4Mx1 DRAMs in 32M byte increments in the most significant 3 memory banks. The low order bank for both PVAX2 and PMariah is fixed at 8M bytes. Mixing 8M bytes memory banks with 32M bytes memory banks is allowed on the PMariah system, but the 32M byte banks must be in the high order memory banks.

With 254kx4 DRAMs, 2M bytes memory banks with a maximum memory range of 8M bytes are supported for an extremely low cost system. Byte Parity is supported with 8 1Mx1 DRAMs for the whole 8M byte address range.

The data access sizes are: byte, word, longword, quadword, and octaword. The DRAMs and VRAMs are 100 nsec access time fast page mode devices.

The video controller is programmable and supports multiple screen resolutions. It is capable of supporting formats up to a resolution of 2048x2048 (dependant on the final ASIC AC timing). The following frame buffers are planned:

Module Type:	Resolution	8 Plane	4 Plane	Refresh Rate	Overlays
High Resolution	1280x1024	yes	yes	66 72	No
Low Resolution	1024x 768	yes	no	60	No
	1024x 864	yes	no	60	No
Dual Head	1280x1024	yes	no	66 72	No
Full Color	1280x1024	24 Planes		66 72	Yes

A 64x64x2 cursor is supported and stored in 1k bytes of off-screen physical memory.

The System ASIC, S-Chip, is a 0.8 micron compacted array with 172k raw gates and ~40%-50% utilization which gives ~75k gates available. The Graphics, Memory, and Video subsystems implementations are ~48k gates.

Figure 1: PVAX2 Block Diagram

Figure 2: Low Cost Graphics/Memory/Video Controller Block Diagram

1.1 MV_DAL Option

The S-Chip MV_DAL option interface supports both Program I/O and LCG Virtual DMA. The protocal is specified in here in a later section. The Cursor<7:0> pins are used for communicating with the MV_DAL Option. The MV_DAL is a RAS/CAS memory bus.

2 Overview

The following sections will give overviews for the major components of the Low Cost Graphics/Memory/Video subsystem in the PVAX2/PMariah System chip, S-Chip. The memory controller is integrated with the graphics and video control. The memory controller supports the memory access modes and special signals required for VRAM and 2D graphics primitives. The graphics address generator operates on both Frame Buffer VRAM and Virtual Main Memory at memroy speeds. Data can be moved to and from VRAM and main memory for off-screen pixel maps and fonts, also at memory speeds. The design is highly pipelined with separate graphics and memory state machines for the highest memory bandwidth efficiency and minimal memory hogging by the graphics controller.

The LCG functionality off loads the CPU and conserves the VAX VUPS for the DECwindows server, operating system, and applications as much as possible. The goal is competitive system performance at minimal cost.

2.1 Memory Request Prioritiy

The low cost graphics controller is last on the priority list. It is effectively a background task. When graphics primitives are present, then it takes up all spare memory cycles. The priority of requests serviced are:

1. Video Controller:
    1. Video Shift Register Load
    2. Cursor Buffer Load - Octaword read = 64 x 2 planes
    3. Memory Refresh
2. NI Controller
3. CPU Interface / Bus Adapter
4. SCSI Controller
5. Graphics Address Generator

2.2 LCG Interrupt Vectors:

The LCG subsystem generates 2 different interrupt signals and status longwords with the following priority:

1. Graphics and Command FIFO and Vertical Retrace
2. MV_DAL Option

2.3 PVAX2/PMariah Low Cost Graphics, LCG, Features List

- LCG is Highly Integrated with the PVAX2/PMariah Memory Controller.
  Graphics is a Background Task - Uses unused Memory Cycles.
- LCG is a Joint Design with VMS DECWindows Engineering.
  Designed for Best Fit with X and Avoids Special Cases
  Works with Virtual Memory:
  - Unlimited Dawing Space
  - Improves Performance
  - Avoids Software Complexity
- Video Modules Designed for Minimal Cost
  Graphics Controller is free. Minimal VRAMs _& Video Out Only

Display	No of VRAMs	VRAM Cost	Cost FY91	Cost FY92
2M Pixel 1 Plane All Resolutions	4 128kx8s	$ 60/$ 36	~$180	~$164
1M Pixel 8 Plane 1024x 864, 1024x 768, & 800x 600	8 128kx8s	$120/$ 72	~$240	~$200
2M Pixel 8 Plane 1280x1024	16 256kx4s	$208/$144	~$295	~$240

128kx8 VRAMs @ $15 (FY91) $10 (FY92)
256kx4 VRAMs @ $13 (FY91), $ 9 (FY92)

- 64x64x2 Cursor Support for both Color and Monochrome
- PVAX2/PMariah LCG supports 2D Graphics Primitives:
  LCG Offloads CPU - Saves VUPs for Server and Applications
  LCG Graphics Primitives Supported:
  - Lines or Vectors
  - 3 Operand RasterOps, Spans, and Points with
    Tiling, Stipples, Color Expansion, & Plane Extraction
  - Text, Both Solid & Stenciled with Variable Sized Glyphs
  - Multiple Clipping Rectangle Support for all Primitives
  - All Operands can be Virtual

2.4 Graphics Performance Estimates (VMS & X-Server Overhead)

	Video Frame Buffer			Virtual Pixel Map	
	8 Plane	24 Plane	1 Plane	8 Plane	1 Plane
Random Vectors Per Second (Worst Case Orientation)					
10 pixel/LCG-30ns	250k	85k	250k	150k	150k
10 pixel/LCG-35ns	200k	200k	200k	125k	125k
10 pixel/PVAX1 SP	240k	N.A.	N.A.	N.A.	N.A.
10 pixel/PMAX	50k	?	?	50k	?
10 pixel/Dragon	35k	N.A.	N.A.	N.A.	N.A.
Rectangle Fill as Million Pixels Per Second					
LCG-30ns	40	13	120	35	100
LCG-35ns	32	10	105	28	80
PVAX1 ScanProc	40	N.A.	N.A.	N.A.	N.A.
Dragon	8	N.A.	N.A.	N.A.	N.A.
Move Area/Scroll as Million Pixels Per Second					
LCG-30ns	18	6	60	15	50
LCG-35ns	15	5	52	12	40
PVAX1 ScanProc	20	N.A.	N.A.	N.A.	N.A.
PMAX	4.0/5.3	?	?	4.0/5.3	?
Dragon	8.0	N.A.	N.A.	N.A.	N.A.
Text as Glyphs/Second					
9x11/LCG-30ns	150k	50k	170k	95k	95k
9x11/LCG-35ns	130k	40k	150k	85k	85k
9x11/PVAX1 SP	170k	N.A.	N.A.	N.A.	N.A.
9x11/Dragon	25k	N.A.	N.A.	N.A.	N.A.

N.A. = Not Applicable

Need the graphics performance for managing large off-screen images for both Scan Proc and Dragon. The bottleneck here will be swapping the image in and out of the frame buffer, even when updating a relatively few pixels, such as a vertical line.

3 PVAX2/PMariah Memory System

The memory controller consists of a state machine, multiplexers, and parity generation and checking. The types of operations supported are reads, writes (VRAM bit masked and non-masked), and read-modify-writes with naturally aligned data sizes: longword, quadword, and octaword. All graphics primitives are broken up into one of these operations and presented to the memory controller.

3.1 Memory Controller State Machine

This state machine controls all of the Memory address, control, and data, as well as, the MV_DAL timing. The Memory Controller State Machine receives all requests from the Memory Arbitration Priority Encoder within the Flow Control State Machine. The request will consist of:

- Requester ID (Video, NI, SCSI, CPU, or Graphics)
- Memory Access Type (Read, Write, Read-Modify-Write, Video Shift Register Load (Full and Split), and Memory Refresh)
- Access Size (Longword, Quadword, Octaword with 16 individual Byte Masks)

The Memory Controller State Machine latches the appropriate address. The address bits indicate whether the access is to VRAM, DRAM, ROM, or an internal Register. The Memory Address Output Controller will feed back the memory state to the Memory Controller State Machine which describes the configuration of the memory. The state machine will acknowledge to the appropriate source that the address has been latched so that the master can get ready for generating the next request address.

When idle, the Priority Encoder within the Memory Arbitration will direct the highest priority access information to the Memory Controller and through an open latch directly to the Memory Address Output Controller. The Memory Address Multiplexer will be left in the RAS address selection mode. The Address will be decoded for the appropriate RASEN assertions. With this approach, the Memory State Machine can assert RAS on the very first cycle, knowing that the address set up and address decoding has already taken place. The Memory State Machine sets all of the control/data paths in the idle state as soon as possible at the end of each memory access. The minimum read latency is achieved this way.

The Memory Controller State Machine is clocked every 60 to 80 nsecs, depending on the actual SOC or Mariah cycle time. The Memory Controller sends control to the Memory Address Output Controller which clocks some signals every 30 to 40 nsecs. These signals will contain 2 bits per signal. The least significant bit will be clocked first and the most significant bit second. This scheme allows signal transitions to occur on any 30 to 40 nsec edge.

The state machine also controls the Pixel SLU when data is coming from memory, as well as, when to output it to memory. The parity checker sends the state machine and the requesting memory master 1 error bit.

Figure 3: Memory Controller State Machine Block Diagram

Figure 4: PVAX2/PMariah Memory System Block Diagram

Figure 5: DWT Memory System Block Diagram

3.2 Memory System Implementation Strategy

The memory system is designed to work with 60 nsec cycles. When the system clock is slowed down, obviously the memory slows down. The first number for memory timing below reflects the 60 nsec cycle. The second is the 70 nsec cycle for the SOC.

The conservative implementation is the 2-way interleaved 240/280 nsec memory cycle times with the 120/140 nsec page mode. The implementation will support up to octaword accesses which gives a maximum bandwidth of 38.1/32.1 Mbytes/sec for reads and 44.4/35.7 Mbytes/sec for writes.

The memory system interfaces to the MV_DAL through 5 74F543 latched transceivers. Data read and written to main memory is latched in both directions. Memory is configured as a 64 bit memory with separate parity bits per byte. For quadword and octaword accesses, 2 32 bit banks are accessed in parallel with 1 RASEN asserted.

Read data is latched into the 74F543s as 2 longwords plus parity, then transferred 1 longword plus parity at a time to the S-Chip, LCG. The M-DAL cycle time is 60/70 nsecs. When accessing an octaword, the first quadword transfers to the S-Chip in 120/140 nsec in parallel with the next quadword page mode access. The second quadword transfers across the MV_DAL while the read access is being terminated and the next access is beginning.

Write data is transferred 1 longword plus parity at a time on the MV_DAL, latched into the 74F543s, then output as 2 longwords plus parity to the DRAMs. When writing an octaword, the first quadword plus parity will be valid for 60/70 nsecs. The second quadword is transferred in parallel with the first quadword write and latched into the 74F543s starting 60/70 nsec after the last longword of the first quadword is assembled.

3.3 Memory Arbitration and Flow Control

The memory arbitration logic prioritizes requests and sets up the required information for the memory controller state machine and associated logic so that when one request is finished, the next memory access can occur immediately afterwards.

The memory arbitration and flow control communicates with all memory requesters for accepting and acknowledging requests. It also directs the octaword graphics data buffer, plane mask, and the Pixel SLU from information supplied by incoming requests, the memory controller state machine, and the graphics address generator. The arbitration and flow control also directs the reading and writing of all internal registers.

The memory state machine must also be involved with accessing the optional Scan Proc graphics which is only accessed in a program I/O environment.

3.3.1 Memory Request Prioritiy

The low cost graphics controller is last on the priority list. It is effectively a background task. When graphics primitives are present, then it takes up all spare memory cycles. The priority of requests serviced are:

1. Video Controller:
    - Video Shift Register Load
    - Cursor Buffer Load - Octaword read = 64 x 2 planes
    - Memory Refresh

2. NI Controller
3. CPU Port
4. SCSI Controller
5. Graphics Address Generator

3.4 Memory Address Output Controller

The Memory Address Output multiplexer selects between the Address In bus, Address Generator, and Video Address for shift register loads and memory refreshes. This section also outputs the address and control signals needed to control both main memory and the frame buffer.

The physical addresses will be multiplexed to output latches. The multiplexing will take care of the different configured memory components.

RAM Configuration	Address Bits
4Mx1 bit DRAMs	11
1M bit DRAMs	10
256kx4s VRAMs	9
128kx8s VRAMs	9

This section also contains a control register which signals the Memory Control State Machine of the configuration and memory type: DRAM or VRAM, as well as, whether the frame buffer is color or monochrome.

There is a 1 bit counter used for the least significant bit of the CAS address for incrementing the counter for page mode accesses: octaword.

All Memory Error and Configuration Registers reside in the section.

3.5 CPU Write Buffering Mechanism for LCG

The CPU write buffers up to a quadword for writes to main memory, the Command FIFO, and the frame buffer.

The Command FIFO is is mapped to a 256k byte I/O address space: 20180000 to 201FFFFF.

The Frame Buffer address space is mapped as 32M bytes: 21000000 to 22FFFFFF. The bottom 16M bytes, 21000000 to 21FFFFFF, are CPU write buffered and the top 16M bytes, 22000000 to 22FFFFFF, are treated as I/O writes which flush the write buffer. LCG Register, 20100000 to 2013FFFF, writes are also considered I/O writes and also flush the write buffer.

3.5.1 LCG Command FIFO Mechanism

The Graphics interface is Command Packet oriented, versus a typical register address interface. The Command Packets contain an Opcode, Opcode specific flags, and parameters. The first longword of a command also contains 2 bits which indicate the number of additional longwords used for this command packet. The command opcode is be decoded for determining which registers are to be loaded.

The benefit of this FIFO mechanism is that since it takes between 1 and 4 longwords per command packet and up to 3 command packets per graphics operation, writing the LCG Command FIFO is a major memory bandwidth consideration.

When the Graphics System is busy, CPU writes to the Command FIFO are directed to the Command FIFO ring buffer in main memory. The CPU interface supplies the access size, byte mask, and within an octaword address<3:2>. The Command FIFO supplies the octaword address.

The LCG Command FIFO always reads an octaword of data from memory. If all longwords are required, then the appropriate LCG registers are loaded as the data is read from memory. The LCG Command FIFO read buffer is 3 longwords in size and it stores the unused longwords. These residue longwords are used on the next command and are not re-read from memory.

When the Command FIFO read buffer is empty and the Address Generator is busy (performing the graphics primitive), then the Command FIFO waits until the Busy signal is deasserted before posting the next memory read. The Busy signal deasserts before it finishes its last memory access, but after being granted a successful memory reference. This timing makes sure that the Command FIFO receives the next memory access cycle, depending on its priority level (last other than the LCG Address Generator).

When ever the Clip List is disabled, the LCG Command FIFO is empty, and LCG AG is Not Busy, then data destined for the Command FIFO is written to the LCG internal registers. Only the longwords required for a complete graphics primitive are written to the internal registers. The remaining longwords are stored within the Command FIFO Residue Buffer. If LCG AG is busy and there is room in the residue buffer, then data is stored in the Residue Buffer. Otherwise, the data is written to memory. This mechanism is called "short circuit".

The clipping rectangle hardware will allow the hardware to clip the drawing primitives to overlapping windows. A clip list can be specified in lieu of immediate clip data. When the clip list feature is active, each primitive will be clipped to each rectangle in the list. The clip list can be anywhere in contiguous main memory, but may not span a 64k byte boundary.

When the Clip List is enabled, then the data will always be written to the Command FIFO no matter if it is empty and/or the LCG is idle. It will also be stored within the LCG internal registers and/or Residue Buffer if the appropriate conditions dictate. The Command can be read multiple times and clipped to multiple rectangles.

There are programmable threshold interrupts available for both Command FIFO full and empty. This is for tuning the Server/LCG interface for stopping the FIFO writes before overflowing the FIFO and waking up the Server when the FIFO gets to an empty threshold. An interrupt will be sent whenever the threshold is reached and the interrupt enable is asserted.

The Command FIFO ring buffer wraps on it programmed size: 64k, 32k, or 16k bytes. The Clip List is also modulo 64k bytes and if it wraps, then it is considered an error. An interrupt will be sent to the CPU whenever the Clip List wraps.

3.6 Address Generator Interface, Virtual Translation, Graphics Interrupts, & Debug HW

All Address Generator requests and grants go through this section because of the Virtual Translations and debug hardware. Command FIFO, Clip List, Virtual Translation, and Address Generator operand requests are all funneled down to 1 request bit, access type, access size, and an ID tag which is forwarded to the memory arbitrator in the Flow Control state machine. The ID tag is transmitted on the Flow Control Bus during the memory or register access and signals the appropriate master that its request is being serviced.

3.6.1 Virtual Memory

The Virtual Translation supports supports 3 translation buffer entries with the base address of the page table pages. There is 1 set of translation buffer latches for each of the three operands for rasterops. The translation buffer entries are: destination, source, and stencil. With this hardware, LCG can now draw graphics primitives to any scattered portion of main memory. This allows the Pixel Maps to reside on the disk until accessed by the graphics system.

The Virtual Translation mechanism reads its page table entries from the Server's P0 page table. Checking is performed on the Valid Bit for both levels of translation, as well as, the access mode (User Write) and the modify bit (on writes) in the second level translation. A maximum virtual address check is also performed. If any faults are found, then the CPU is interrupted with the appropriate status available.

The performance will degrade, depending on the number of page crossings and the number of times that the physical address must be read from memory. Vertical vectors are the worst case; crossing a page boudary on every access. Horizontal vectors and Rasterops are much better since memory is accessed in contiguous byte accesses.

Neither the Command FIFO nor the Clip List are virtual. They must be contained in contiguous physical memory.

3.6.1.1 Virtual Memory Interface Overview

There are 3 Virtual Address latches for comparison with the address generator's 3 operand addresses. One latch is used for Vectors and Destination operands. A second latch is used for all 2 operand primitives. The third latch is used for the stencil operand. The address generator's addresses are 28 bits wide (bits<1:0> are ignored), but the least significant 7 bits (address<8:2>) are used for within a page. Address<29:09> are required for comparison.

Bits <29:09> of the operand's virtual address is compared with the corresponding Virtual Graphics Operand Address Latch for deciding whether the corresponding Physical Graphics Operand Address latch contains the correct physical address. The comparison is accomplished in 2 parts. Bits <29:15> indicate whether the current Process Page Table Page Address latch contains the correct page address. When this portion of the compare is true, but bits <14:09> fail, then the current Process Page Table Base Address latch contains the correct Page Table Base address. For this case, bits <14:09> are used to index into the Page Table Page for accessing the correct PFN.

When bits <29:15> of the virtual address and bits <20:06> of the Virtual Graphics Address latch comparison fails, then the P0 System Base Address latch is added to bits <29:15> of the virtual address for obtaining the correct Process Page Table Page Base Address. This

results in a double lookup and a length check for accessing the correct PFN. The P0 System Length Latch is also compared against the Operand's Virtual Address.

Whenever a miss occurs and the PFN is successfully loaded, then the Operand's virtual address will be loaded into the corresponding Virtual Graphics Operand Address Comparison Latch. When the Valid bit is deasserted on either access, an interrupt is generated and the virtual address may be accessed by the CPU. All drawing will suspend, until the CPU corrects the situation and restarts the Address Generator.

Cache Coherency is garenteed by the Invalidate TB mechanism. VMS writes an LCG I/O address which flushes all cached translation buffer entries whenever it has potentially modified one of the DECWindow Server's PFNs.

4 Pseudo-code for LCG VM Translations

```
Assume the following constraints on the page table
data structures and the manipulations on them:

1) The system page table is physically resident and
   contiguous.

2) The process P0 page table resides in virtually
   contiguous system address space.  Therefore,
   the system page table entries which map the P0
   page table are physically contigous.

3) The P0 page table is page-aligned.

4) No page of the process P0 page table can become
   non-resident without its constituent PTE's
   first becoming invalid.

5) The system virtual address of the process header
   (i.e. the P0 page table) can be prevented from moving.

6) The process can be prevented from shrinking below
   LCG_P0LR
```

Assume the existence of the following registers ("XX" stands for one of {source,stencil,dest}):

LCG_PA_S_P0BR<26:2>

Physical address of the system page table entry which maps the first page of the process P0 page table.

The SPTE pointed at by LCG_PA_S_P0BR maps the address contained in the VAX P0BR register.

LCG_P0LR<29:9>

P0 length register.
The P0 virtual address of the highest mapped page.

LCG_NEXT_VA<29:0>

The next address to be accessed by LCG.
This will be the virtual address to be translated if the operand is virtual.

LCG_XX_PA_P0PTP<26:9>

For each of the 3 LCG operands: the physical address of CURRENT P0 page table page. That is, the address of the page table page containing the PTE for LCG_XX_VPN.

LCG_XX_VPN<29:9>

The virtual page number associated with LCG_XX_PFN.
This is the same as <29:9> of the most recently translated address.

LCG_XX_PFN<26:9>

The page frame number associated with LCG_XX_VPN.
In other words, physical address bits <26:9> of the most recently translated address.

Further assume the following symbols which are used below to identify intermediate values.

PA_SPTE

The physical address of the system page table entry which maps the desired P0 page table page.

PA_P0PTE

The physical address of the process page table entry which maps the virtual address.

LCG_NEXT_PA

The translated address corresponding to LCG_NEXT_VA.

SETUP:

The following registers must be loaded by the server:

LCG_PA_S_P0BR must be loaded with the physical address of the SPTE which maps the first page of P0PT.

LCG_P0LR must be loaded with the highest virtual address which is to be accessed by LCG.

```
HARDWARE ALGORITHM

 The following assumes maximum width physical addresses.
 (Ignoring that pv2 only has 27 bits of physical address space )

 /* TB-hit case */

   if (LCG_NEXT_VA<29:9> == LCG_XX_VPN<29:9>) {
      go to HIT;
      }

 /* Check for length error - status bit from the Address Generator */

   if (LCG_NEXT_VA<29:9> > LCG_P0LR<29:9>) {
      interrupt_and_halt(length_error)
      }

! VPN load moved to SINGLE_MISS flow below

 /* Single miss: desired PTE is in same page table page */

   else if (LCG_NEXT_VA<29:16> == LCG_XX_VPN<29:16>) {
       LCG_XX_VPN<29:9> = LCG_NEXT_VA<29:9>;
       goto SINGLE_MISS;
       }

 /* Need new P0 page table page - go back through system page table */

 else {
    LCG_XX_VPN<29:9> = LCG_NEXT_VA<29:9>;
    PA_SPTE<29:2> = LCG_PA_S_P0BR<29:2> + LCG_XX_VPN<29:16>; /* AG */
    SPTE = *PA_SPTE;

! prot/mod checks removed

    if (SPTE<31:31> <> 1) {
       interrupt_and_halt(translation_not_valid);
       }

    LCG_XX_PA_P0PTP<29:9> = SPTE<20:0>;

SINGLE_MISS:

    PA_P0PTE<29:9> = LCG_XX_PA_P0PTP<29:9>;
    PA_P0PTE<8:2>  = LCG_XX_VPN<15:9>;
    P0PTE<31:0>    = *PA_P0PTE;
    if (P0PTE<31:31> <> 1) {
       interrupt_and_halt(translation_not_valid);
       }
    else if (P0PTE<PROT> <> USER_WRITE) {
            interrupt_and_halt(protection_violation);
            }
    else if (P0PTE<M> == 0  AND  operand_is_destination) {
            interrupt_and_halt(set_modify_bit)
            }

    LCG_XX_PFN<29:9> = P0PTE<20:0>;
HIT:
    LCG_NEXT_PA<29:9> = LCG_XX_PFN<29:9>;
    LCG_NEXT_PA<8:0>  = LCG_NEXT_VA<8:0>;
    }
```

Figure 6: Command FIFO, Clip List, and Virtual Translation Block Diagram

5 FIFO OPERATION

)

5.1 INTRODUCTION

This section describes functionality and timing of the FIFO/CLIPLIST controller. LCG is fed command packets by the CPU. LW and QW writes to a specific address signify LCG command packets. Since the command packets are sent to the LCG at a pseudo random fashion, for synchronization a (programmable size) FIFO is implemented in the physical memory (MEMORY FIFO). LCG can receive commands from CPU directly or it can read commands from two different streams; namely the MEMORY FIFO and the CLIPLIST. The CLIPLIST is an alternate source of commands that is used in conjunction with (possibly multiple) DRAW-UNITS (Figure 1). The CLIPLIST also resides in physical memory. In addition to the FIFO and CLIPLIST There are three registers which are used to store the unexecuted LWs. These three registers are collectively called the RESIDUE BUFFER. In addition FIFO controller has the following registers:

- FIFO BASE: Most significant 11 bits of the FIFO address.
- FIFO HEAD: Least significant 14 bits of FIFO head address.
- FIFO TAIL: Least significant 14 bits of FIFO tail address.
- CLIP BASE: Most significant 11 bits of the CLIPLIST address.
- CLIP HEAD: Least significant 14 bits of CLIPLIST address.
- FIFO MASK: specify size of the FIFO and threshold for EMPTY and FULL.
- FIFO SAVE: Return address for FIFO HEAD.
- CLIP SAVE: Return address for CLIPLIST HEAD.

Figure 7: Command FIFO & Clip List Data Flow Diagram

5.2 BASIC OPERATION

The FIFO/CLIPLIST controller does simple traversal of MULTI-LW LCG command packets.CPUC writes (LW or QW) to the FIFO address are signaled to the FIFO controller. Two two bit counters (WF_CNT and WCNT) define the basic state of the controller. At reset both counters are zero. When CPU writes (LWs or QWs) into the FIFO, WF_CNT is incremented by one for every LW detected on the PXDAT. Bits <23:22> of the First LW of a command packet (packets are 1 to 4 LWs) specify size of the packet. These bits are loaded into the WCNT. Every LW appearing on the PXDAT will cause WCNT to be decremented by one. When all the LWs of a packet have appeared on the PXDAT, WCNT is back to zero. This way the controller knows whenever WCNT is zero it should be expecting the next LW appearing on the PXDAT to be the first LW of a packet. Another two bit counter (CFCNT) is used to generate LW numbers for the rest of LCG. On the first LW of a packet CFCNT is reset to zero every following LW appearing on the PXDAT will cause CFCNT to be incremented by one until the first LW of the next packet will reset CFCNT. FIFO controller keeps track of the following information:

- LONGWORD COUNT
- SHORT CIRCUIT
- AGBUSY
- CLIP_WALKER
- DRAW-UNIT
- RESIDUE BUFFER VALID LWs
- FULL and EMPTY
- FIFO MA_REQ BITS
- COMMAND STREAM (FIFO/CLIPLIST)

FIFO controller generates the following status/interrupt bits:

- CLIP WRAP
- FIFO RANGE LOAD/ERROR
- COMMAND FIFO STALL
- PACKET BREAKPOINT
- EMPTY AND FULL INTERRUPTS
- TOGGLE INTERRUPTS

5.2.1 SHORT CIRCUIT

FIFO/CLIPLIST controller works on an OCTAWORD basis. Starting on an OCTAWORD boundary the controller decides if the LW has to be immediately executed as a part of a command packet by the LCG or be stored to be executed later. At every instance the FIFO controller has to calculate a logic equation to find if it can put the next CPU write intended for FIFO into the RESIDUE BUFFER or not. Short circuit is referred to a situation in which there is still room in the residue buffer for more LWs. The advantage of SCIRCUIT

is that while in short circuit any and all the command packets coming in will be executed immediately or stored in the residue buffer to be executed after the ADDRESS GENERATOR is finished with the packet it is currently executing.

Once the SCIRCUIT is de-asserted all the writes to FIFO will be written into the FIFO which resides in physical memory. The FIFO/CLIPLIST controller will keep writing into the FIFO while FIFO packets are coming form the CPU side. Once there is more than ONE full OCTAWORD in the FIFO the controller asserts its request signal to read an OCTAWORD from the FIFO. Two up-counters help the controller keep track of the HEAD and TAIL of the FIFO. When the FIFO controller is granted to read an OCTAWORD, it traverses the FOUR LWs read. Through this process the FIFO will act as an intermediate buffer for LCG command. As mentioned FIFO controller uses SCIRCUIT to decide if the LCG packets (CPU writes) will be executed immediately or will be store in the FIFO. SCIRCUIT will be used by the MEMORY controller to determine the FCTL control bits on SOC writes to FIFO (refer to FCTL definition).

5.2.2 AGBUSY

In order to sequence packet traversal a signal (AGMREQ6) is generated by the AG which indicates that it is in the process of doing an action packet. Action packets are those which require other sections of LCG to wait until AG is done. In the case when LCG is executing command packets in single step mode the situation is identically equal to AG being busy. And finally after a WAIT_FOR_SYNCH packet is executed the situation is the same. In all these cases any packet showing up on the PXDAT will not be executed buy the LCG. Command packets coming in will be stored either in RESIDUE BUFFER or in the FIFO. A composite signal (AGBUSY) is used to signify this condition. AGBUSY is used in SCIRCUIT calculation.

5.2.3 DRAW UNIT

Since it is required that a collection of commands, usually drawing packets, collectively called a DRAW-UNIT be executed by the LCG against a number of CLIP-LISTs each called a CLIP-UNIT, FIFO controller needs to have a save mechanism to go back to the beginning of a DRAW-UNIT. The same kind of mechanism is also provided for CLIP-UNITs. FIFO controller keeps thrack of state of DRAW-UNITs using a flop that if/when SET, implies that the FIFO controller is either in a DRAW-UNIT or JUST STARTED a DRAW-UNIT. The flag IDU (IN-DRAW-UNIT) is used to calculate SCIRCUIT at the boundaries of OCTA WORDs.

5.2.4 BASIC TIMING

FIFO controller latches FCTL<16:0> during CLKP4. PXDAT<31:24,19:16> are decoded to decide if any of the FIFO controller registers must be loaded from PXDAT. All the register loads happen either during CLKP2 or clocked at the falling edge of clkp2. SCIRCUIT calculation is the most timing sensitive signal and is calculated between rising edge of CLKP4 and falling edge of CLKP1. SCIRCUIT is clocked at the falling edge of CLKP1 at the OCTA WORD boundaries or whenever any of the registers effecting SCIRCUIT are loaded from PXDAT due to either register writes or any packet which sets any of these registers.

This scheme for SCIRCUIT timing is dictated by the fact that FCTL changes its value sometime after CLKP3 rising. PXDAT is available sometime during CLKP4. In the case of RESBUFF reads PXDAT as seen by the FIFO controller is very close to CLKP1 rising. Decoding of 11 bits of PXDAT and then SCIRCUIT calculation logic has to be done in the time between PXDAT becoming valid sometime before CLKP1 falling. This Guarenttes that the MEMORY CONTROLLER will have enough time to respond when the next packet come in from the CPU.

5.2.5 CLIP-Wlaker and TOGGLES

Some LCG command packets can turn on the CLIP_WLAKER mechanism in the FIFO/CLIP controller. This mechanism allows LCG to execute commands form two steams in the memory namely FIFO and CLIP LIST. The contorller detects its directives to read from the FIFO or CLIP streams by reading THREE flag bits on the first LW of some specific commands. These three (SAVE-RESTORE-TOGGLE) bits fully specify what the FIFO/CLIP controller is to do next (table 1).

Table 1: FIFO/CLIP SAVE-RESTORE-TOGGLE

Func Bits	Cur FIFO	Cur IDU	SF	RF	SC	RC	Next Fifo	Next IDU
B_DU (111)	1	-	1	0	0	0	0	1
B_CLIP (111)	0	-	0	0	1	0	0	SAME
E_DU (101)	1	1	0	1	0	0	0	SAME(1)
E_DU (101)	1	0	0	0	0	0	1	SAME(0)
E_CLIP_LIST (110)	0	-	0	[illegible]	0	1	1	0
T_DU (011)	1	1	0	1	0	0	0	SAME(1)
T_DU (011)	1	0	1	0	0	0	0	1
B_DU_NO_T (110)	1	-	1	0	0	0	1	1
E_DU_NO_T (100)	1	1	0	1	0	0	1	SAME(1)
E_DU_NO_T (100)	1	0	0	0	0	0	1	SAME(0)
E_CLIP_NO_T (100)	0	-	0	0	0	1	0	0
E_CU/CH_STR (010)	-	-	0	0	0	0	NOT	SAME(0)
CLR_IDU (001)	-	-	0	0	0	0	SAME	0
NOP (000)	-	-	0	0	0	0	SAME	SAME

5.3 Write Mask Latch and Generation Logic

This section contains the graphics Write Plane Mask which is used for enabling the writing of individual planes. This mask is used for both VRAMs, the frame buffer, and DRAMs, main memory.

5.3.1 Address Generator Byte Enables

These signals are the CAS enables which allow partial writes of accesses from a byte to an octaword. These signals are AGMSK<7:0> H and they are asserted no later than on the cycle that the Address Generator's address is valid. The Memory Controller controls a 2 to 1 multiplexer which selects the correnspondingg 8 byte masks for octaword accesses.

5.3.2 Address Generator Plane/Monochrome Bit Masks

The AGWMSK<31:00> H signals are used for plane masking for 8 plane video. The 8 bit Plane Mask Latch is replicated on all 32 bits. These signals are also used for monochrome bit maps. When writing to monochrome bit maps the server will write all ones into the Plane Mask Latch. The Mask Generator will force zeros onto all bits which are not to be modified within bytes to be written. The AGWMSK<31:00> H signals are required for writes the same cycle as the Address Generator Arbitration Signal is asserted. It is used for enabling the VRAM internal plane masking hardware and is needed before RAS is asserted.

The Monochrome frame buffer is 16 bits wide, 2 VRAMs. During Monochrome frame buffer writes, AGWMSK<15:00> H are the only signals used. If the mask spans lies across a natural longword, then either a read-modify-write or 2 word writes must be performed. It turns out that 2 word writes take 8 cycles and 1 longword read-modify-write takes 11 cycles. The read-modify-write will use all 32 bits and control the PIXEL SLU's DST OUT multiplexer to select on a bit by bit basis the source or destination data.

When writing to monochrome pixel maps, the Plane Mask is used to indicate which pixels are written and the mask generator will create masks on the bit boundaries. There are 4 cases:

- left edge asserted
- right edge asserted
- middle bit(s) asserted
- all bits asserted

5.4 Graphics Data Buffer

The Graphics Data Buffer is used for an intermediate storage for the Source operand of RasterOps. The RasterOp primitive is used for the majority of X (DECwindows) primitives, as well as, DMA of main memory to and from frame buffer and MV_DAL options.

The size of the Data Buffer needs to be worse case twice the size of the destination access size for the Source operand for RasterOps which is 2 octawords.

The loading of the Graphics Data Buffer is controlled by the Memory Arbitration and Flow Control section. The outputting is controlled by the Pixel SLU. The Graphics Data Buffer and Pixel SLU are partially de-coupled from the Address Generator. The Address Generator is calculating the next address and performing window clipping while the current data is operated on.

The output multiplexer is used as the first stage of the funnel shifter. The output multiplexer is made up of 2 16 bit 8 to 1 multiplexers with individual selects.

5.4.1 Plane Extraction

Plane Extraction is performed here. There is a 4 bit 8 to 1 multiplexer which connects to each bit of the 32 bit Pixel Data Bus. Any bit position out of every byte can be selected and loaded 4 bits at a time into a 32 bit latch. It takes 8 longwords to assemble a 32 bit compacted longword. This is used for extracting a plane from a color pixel map and compacting it into a contiguous bit map.

Figure 8: Graphics Data Buffer Block Diagram

Figure 9: Pixel Shifter and Logic Unit (SLU) Block Diagram

5.5 Pixel Shifter and Logic Unit (SLU)

The Pixel SLU, Shift and Logical Unit, consists of a 32 bit logical unit which executes all 16 boolean functions with the X-Window encoding, 32 to 16 bit funnel shifter, and all the data paths required for all rasterops and vector functions. All data reads and writes from and to system memory and the frame buffer come through this data path. This data path is pipelined for allowing a through-put of 1 longword per cycle, 60/70 nsec. The source paths of this data path are reserved for the address generator. The destination path never has intermediate data stored within it between memory accesses. Data which is both read and written from the destination will be performed with a read-modify-write operation.

5.5.1 Pixel SLU Source Data Path

The Graphics Data Buffer is be used as source 1 for rasterops.

The rasterop source 1 input has a 32 to 16 bit funnel shifter, which can be used in 2 modes. The first mode is used when shifting data as pixels without color expansion. The shifting in this case is controlled by the shift count and the size of the source data.

The other mode is used for color expansion, where 2 bits will be used to select either the foreground or background color latches. The color expansion mode shifts 2 bits at a time. The least significant 2 bits select the select either the foreground or background pixel.

Transparent text, tiles, and stipples are supported with or without a Stencil operand. Read-modify-writes will be used whenever either the Stencil operand or transparent mode is enabled. Read-modify-writes will always be used to main memory, unless the destination operand is 8 planes and the plane mask is fully asserted.

The source side of the logical unit is fed by a 3 to 1 multiplexer. Two of the inputs are the foreground and background color registers. The output of the funnel shifter is also an input to this multiplexer.

5.5.1.1 Pixel SLU Source Funnel Shifting Description

Tiles, not color expanded, and stipples, color expanded, can be loaded into the Graphics Data Buffer. The tile or stipple must be padded by an access size and must be at least 1 access size in length. Exact 16 pixel Tiles and 32 pixel Stipples are special cased and only read the data once. All other Tiles and Stipples read the data multiple times from memory. Only Tile and Stipple strips are supported. Tiles and Stipples must be access size aligned. Full window tiles and stipples are not supported. The Destination operand must have the same number of scan lines as the tile or stipple.

The funnel shifting can occur over arbitrary sized source data from memory. The Graphics Data Buffer contains 8 longwords; 4 for the octaword destination access and 4 for the residue. The output multiplexer of the buffer is used for selecting the appropriate byte from each longword. The first stage of the funnel shifter is 3 8 bit 4 to 1 multiplexers which can shuffle the bytes to the appropriate position for the bit shifter which follows. The bytes must be shuffled for both right to left and left to right shifting. The bit shifter shifts from left to right 0 to 7 bits. It is made up of 16 1 bit 8 to 1 multiplexers. The bit shifter is used for both monochrome and color expansion.

The output of the funnel shifter is registered and is clocked every SOC cycle or Half Cycle (30 to 40ns). The output selects for the Graphics Data Buffer's output multiplexer are also clocked with the Half Cycle clock. The 2 32 bit Source Output latches are latched a word at a time every Half Cycle. The data is accumulated for merging with the 32 bit Destination Data on Read-Modify-Writes or written directly to memory. This shifting mechanism can provide a longword every cycle through the Logic Unit.

5.5.2 Pixel SLU Destination Data Path

The destination path is used for all reads and writes to memory from every potential memory master. The address generator never uses the latches within this data path as temporary storage.

There are 2 32 bit latches at the output of the Pixel Logical Unit. The output of these latches are multiplexed with the Address Generator Write Plane Mask by a 3 to 1 32 bit multiplexer. The output of the multiplexer feeds the memory output register which drives the MV-Dal.

5.5.3 Pixel SLU Logical Unit and Output Data Path

The Logical Unit is implemented as a 4 to 1 multiplexer, where the opcode is the data inputs and the selects are the source and destination data. The function code for the Logical Unit is controlled by the Pixel Function latch during drawing operations and the Flow Control during any other memory access.

The output of the Logical Unit feeds into into 2 32 bit Destination latches. The Destination latches are written once per cycle.

The logical unit 2 to 1 output multiplexer selects between the logical unit and the destination data. The selection between the logical unit or the destination can be performed on the bit basis, where the mask logic controls the selection.

5.5.4 Stencil (Source 2) Operand

The Stencil operand is used as the masking operand for 3 operand RasterOps. It will always be a contiguous bit plane. It can either be expanded 1 bit to a color pixel or used as contiguous bits for monochrome bit maps. The Stencil operand is restricted to a longword latch plus a longword residue latch and must be accesses 1 longword at a time.

The Stencil operand is read before the Source operand. The input of the funnel shifter is fed by either the Graphics Data Buffer or the Stencil operand. The Stencil operand has 1 longword residue latch before this multiplexer and 1 mask latch at the output of the funnel shifter. The Stencil operand is fed through the funnel shifter for aligning with the Destination operand.

5.5.5 Masking Logic

The masking logic provides 3 functions. One function provides the plane mask for writing to main memory, DRAM, without the write per bit capability. The second function allows the Source 2 operand for rasterops to be logically ANDed with the plane mask and provide the bit by bit selection. The third mode uses the output of the funnel shifter to also be logically ANDed with both the plane mask and the Source 2 operand and then provide the bit by bit selection. This mode supports both color expansion and no color expansion. The third

mode is used for transparent text, tiles, and stipples, where the deasserted bits allow the destination to pass unmodified.

The masking logic as a whole is forced to all ones for normal memory operations. During drawing operations, both the Stencil and the Transparent inputs are separately forced to ones when their function is disabled.

5.5.6 Pixel SLU Control

The control of the Pixel SLU is determined by a control register and is directed by the memory arbitration and flow control. The graphics address generator provides control status to the memory flow control for directing the Pixel SLU along with the octaword block data buffer and VRAM write mask.

Some of the functions below do not require a read-modify-write to VRAM based destination data. These functions will be special cased for the added performance improvement. They are marked with (*) below.

5.5.7 X Windows Boolean Function Codes

```
* GXclear              0x0        --  set to 0's
  GXand                0x1        --  src AND dst
  GXandReverse         0x2        --  src AND NOT dst
* GXcopy               0x3        --  src
  GXandInverted        0x4        --  NOT src AND dst
  GXnoop               0x5        --  dst
  GXxor                0x6        --  src XOR dst
  GXor                 0x7        --  src OR dst
  GXnor                0x8        --  NOT src AND NOT dst
  GXequiv              0x9        --  NOT src XOR dst
  GXinvert             0xa        --  NOT dst
  GXorReverse          0xb        --  src OR NOT dst
* GXcopyInverted       0xc        --  NOT src
  GXorInverted         0xd        --  NOT src OR dst
  GXnand               0xe        --  NOT src OR NOT dst
* GXset                0xf        --  set to 1's
```

Figure 10: Address Generator Block Diagram

5.6 Address Generator

See LCG ADDRESS GENERATOR DESIGN SPECIFICATION for a detailed description.

The Address Generator supports lines with a modified Bresenham algorithm, as well as, scrolling, rasterops, text, solid colored or stippled rectangles, tile and stipple strips, scan line spans, and DMA to and from the frame buffer, main memory, and Scan Proc Graphics. All primitives other than vectors are supported with the RasterOp command packets and are specific uses of the general RasterOp hardware.

The Address Generator communicates with the Memory Control State Machine and Memory Arbitration and Flow Control through the Memory Address Output Controller. The Address Generator sends the physical or virtual memory address directly to the Memory Address Output Control with a 2 bit code which indicates whether it is a physical address or 1 of 3 Virtual Addresses: DST, SRC1, or SRC2. The Memory Address Output Controller takes care of the virtual to physical translations and refilling the PTEs on misses.

5.6.1 Operand Addressing

The Address Generator supports 3 operands and the associated registers and latches to support addressing these operands.

There is a Base Address latch for each operand. Each is 28 bits and contains the linear longword address of the first pixel of the operand on the desired scan line.

A Y Step latch is required for each operand. These latches contain the 14 bit 2s complement number for obtaining the linear longword address of the next scan line for each operand. The value contained in the Y Step latch is added to the Base Address latch and the result is then stored in the Base Address latch.

The 16 bit 2s complement X Position register keeps track of the X coordinate in pixels of the destination operand, relative to the left edge of the Destination Base Address. A 16 bit Initial X Position latch is used for storing the first X position needed for each scan line of a RasterOp.

There is a 16 bit 2s complement Bias latch for each operand which contains the pixel bias which when added to the X Position gives the the current address for each operand relative to its Base Address. The Bias value plus the X Position is shifted right 3 bits for monochrome before being added to the Base Address latch for finding the correct byte address. For color the Bias value plus X Position is not shifted before being added to the Base Address.

The result of these 2 adds is then stored in the Next Address latch which is sent to the Memory Address Output Controller. The Memory Address Output Controller sends it directly to the physical address multiplexer when the Next Address is physical. When the address is virtual, it must first translate it from a virtual address to a physical address before sending it to the physical address multiplexer.

5.6.1.1 RasterOp Addressing

In a multi-operand rop, corresponding pixels in the respective operands live at different X coordinates in their respective address spaces. Furthermore, the drawables associated with each operand may have different byte or word alignment.

A mechanism is needed to normalize the three operands to a single X address space relative to the Memories in which the operands reside.

This is accomplished by providing a pixel bias value for each operand. The pixel bias is "the number to add to the current X drawing coordinate to find the pixel address" for each operand. Software sets up these bias values, taking into account both memory alignment and the difference between the source and dest x offsets.

```
MONOCHROME EXAMPLE

  Assume we have operand SRC1 with the following attributes:

  BASE ADDRESS OF PIXEL 0,0:      1000
  Bit offset of pixel 0,0:          05
  Y step:                          200

  Assume we have operand DEST with the following attributes:

  Base address of pixel 0,0:      2000
  Bit offset of pixel 0,0:          01
  Y step:                          500

  We are asked to do the following copy:

  SRC1 x:                          100
  SRC1 y:                          100

  DEST x:                           50
  DEST y:                           50

  WIDTH:                            70
  HEIGHT:                           80

  The registers are loaded as follows:

  SRC byte address = (SRC1 X BIAS + X DRAW OFFSET)<16:3> + SRC Y BASE
  DEST byte address = (DEST X BIAS + X DRAW OFFSET)<16:3> + DEST Y BASE
```

5.6.2 Rasterops

The RasterOp hardware described here is used for all graphics drawing primitives other than Vectors. The hardware is very flexible and has been jointly specified by this hardware group and DECwindows/VMS. One, two, and three operand RasterOps are supported to the Frame Buffer and Virtual Memory. The 16 X-Window boolean operators are supported.

Both tile and stipple strips are supported. Strip implies that the source and destination must have the same number of scan lines. Arbitrary sized tiles and strips are supported. The restrictions are that the source must be or expand to be at least 1 octaword, 16 bytes, in length and it must be octaword aligned. The tile or stipple can be padded with multiple complete copies. Odd sized tiles and stipples will force the destination operand to be accessed twice when the tile or stipple ends and is then read again.

Stipples can either be a contiguous bit stream or a bit per pixel bit stream. The Source 1 Plane Index specifies which plane is to be used. Transparent stipples are supported. The planes which select the Foreground and Background latches are routed to the masking logic for selecting the Destination operand when ever the bit is clear. The stipple bits will be logically ANDed with the Write Plane Mask and Source 2 operand if enabled. The resulting mask will select between the Logical Unit and the Destination operand at the bit level for main memory accesses.

RasterOp Span and Continue Span command packets are supported for the complex operations. A span is the RasterOp function for a scan line. Trapezoids and tiled or stippled vectors are accomplished with RasterOp Span and Continue Span command packets. The Continue Span command packet directs the Address Generator to update the Base latches with the next scan line linear longword address and supplies a new X Position and X Count for the destination operand. The number of source operands are specified with each command packet. The source operands are already set up before the first RasterOp Span packet and are re-initialized for each scan line by the address generator.

The addressing for each operand is described above in the Operand Addressing section. All registers and latches described are used when executing 3 operand RasterOps.

The Write Plane Mask specifies the planes which are to be modified when writing to the destination operand. The Source 1 Plane Index latch specify the plane of interest for single plane per color pixel mode. This can be used for both plane per pixel stipples and moving bit maps between planes.

The Source 2 operand must be fetched before the Source 1 operand. The input of the funnel shifter is fed by either the Data Buffer or the Source 2 operand. The Source 2 operand has 1 longword residue latch before this multiplexer and 1 mask latch at the output of the funnel shifter. The Source 2 operand is fed through the funnel shifter for aligning with the Destination operand.

A quadword is required to result in a longword of Source 2 for masking. When used for color, 4 bits will be used for each longword and 16 bits for an octaword. A longword of the Source 2 operand will be fetched every 2 Destination accesses for color pixels.

When the Destination is a contiguous single plane operand, monochrome bit map, then the Destination accesses will be limited to a single longword, 32 pixels, and a longword of the Source 2 operand will be fetched between every Destination access.

The Address Generator requests an octaword for the Source 1 data and loads it into the Data Buffer through the Pixel SLU destination path. When the data is read from main memory, the data is checked for correct ECC and corrected. The Data Buffer is funnel shifted down to 16 bits for being operated on in the Pixel SLU with the destination data. The first access on a scan line will access 2 octawords for always having the correct amount of data for 1 octaword destination access.

The shift amount is determined by the source address, source plane index, and destination address. The beginning of the source data must be aligned with the beginning of the destination. The source must also be wrapped when it is smaller than the destination. The x dimension must be wrapped for tiles and stipples. The wrapping occurs by reading the source a second time and performing 2 operations on the same destination data with the appropriate masking applied.

Funnel shifting is accomplished by having the output multiplexer of the Data Buffer used for selecting the appropriate byte from each longword. The first stage of the funnel shifter shuffles the bytes to the appropriate position for the bit shifter which follows. The bytes must be shuffled for both right to left and left to right shifting. The bit shifter shifts from left to right 0 to 7 bits. This works for both monochrome and 8 plane color. It also allows planes to be moved.

The Pixel logical unit is a 16 bit logical unit. Data is pipelined and control is clocked every 1/2 cycle, 30ns, for providing an effective throughput of 1 longword every cycle, 60ns.

The address generator requests a destination octaword read-modify-write or just a write, depending on the boolean function and the destination, DRAM or VRAM. The Pixel SLU's memory data input receives the destination data into a double buffered 32 bit latch. A 16 bit 4 to 1 multiplexer for selecting the appropriate word is fed by the double buffered latch. The multiplexer feeds both the logic unit and the output multiplexer. The logical unit makes 2 passes at the data with shifting both the source and destination data and writing a word into the 32 bit output latch. The output latch is 2 longword latches which act as a double buffered longword latch. The destination latch feeds the ECC generation logic which then is latched again at the ASIC memory data transceivers.

The masking operation to VRAM of the written data is taken care of with the VRAM Write Plane Mask latches. The write mask contains a byte mask for internal bytes and begin and end bytes for the byte boundaries for both edges of the rasterop. The boundary bytes are generated with the instruction of the address generator which contains the address and pixel count. When a third operand is used, the third operand is logically ANDed with the plane mask for generating the correct mask with write only operations. This only works for longword accesses. Accesses which are larger than a longword and have edge masks will be performed with read-modify write operations. When a read-modify-write operation is in process, then the ANDed Mask becomes the multiplexer select between the Logical Unit and the destination data. Deasserted bits will select the destination data. Rasterops to main memory, DRAM, will always perform read-modify-writes for performing the plane masking.

5.6.2.1 Scrolling and Window Moves

Scrolling and window moves are accomplished by using the RasterOp Command Packets and hardware. The typical case described here is a 2 operand RasterOp which requires both the Source 1 and Destination Command packets.

No special case packets are supported since all operations can be either to the screen or to virtual memory. The addressing is described in the Operand Addressing section above. The registers used are: DST and SRC1 Base latches, DST and SRC1 Y Step latches, X Position register, Initial X Position latch, DST and SRC1 X Bias latches, DST X Count, Initial X Count, and Y Count.

The data moves to and from the Data Buffer with octaword accesses. Data moves through the Source 1 path of the Pixel SLU. As the data is moving it can be shifted, inverted, or left alone. The plane mask will be used on writes to the frame buffer, VRAM, with write octawords and octaword read-modify-writes with the plane mask will be used to main memory, DRAM.

5.6.2.2 Text

Text will be taken care of with RasterOp Primitive above. The description below describes the flow for this particular application of the general RasterOp hardware.

Two types of text are supported: transparent (masked) and opaque (unmasked). If Transparent is selected, then all deasserted bits will disable the writing of the appropriate pixels. The asserted bits will be written with the foreground color for color. The mask logic will take care of the disabling of the writes. The font will also be loaded into the third operand of the rasterop data path. This operand will be logically ANDed with the plane mask and select on a bit basis the destination operand or the logical unit. This will work for both 1 and 8 bits per pixel. When opaque characters are used, then the bit selects between

two colors: foreground or background for color. This functionality resides within the Pixel SLU.

The text source data is read from memory, either main memory or VRAM. It is passed through the Pixel SLU and stored into the octaword block data buffer. The octaword buffer then presents the appropriate bytes to the funnel shifter which provides the shifting down to 2 bits for the color selection.

The text can be a linear source operand and not the same shape as the destination. The offset registers and length counters will be used as described above in the rasterop case.

5.6.2.3 Solid Colored or Stippled Rectangles

Rectangle Fill will be taken care of with either a 1 or 2 operand RasterOp Primitive above. The description below describes the flow for this particular application of the general RasterOp hardware.

Solid colored or stippled rectangle fills are accomplished by loading the Data Buffer with an arbitrary pattern and number of bits from either the host (Maximum is 128 bits) or memory. The Data Buffer will act as the source for filling a rectangular portion of either VRAM or main memory. The fill pattern will wrap back on itself on a scan line by scan line basis. The data can either be used directly, color expanded, or as both the source and mask for transparency.

5.6.2.4 DMA

DMA is implemented very much like scrolling above. It is also implemented with the 2 operand RasterOp Command packet. It is not limited to contiguous linear memory. It is typically used for moving pixel maps to and from frame buffer, VRAM, and main memory or moving windows. The Data Buffer is used for temporary storage when reading an octaword. Both the VRAM mask latches and the window clipping may be enabled when writing to VRAM.

5.6.3 Bresenham Vectors

Bresenham vector drawing is supported within the address generator and Pixel SLU.

The address registers/latches are mapped a little differently for vector drawing than described earlier in the Operand Addressing section. Three 16 bit latches are required for the Error Calculations which direct the X and Y steps for drawing the line. The Error Accumulator Register uses the Source 2 Bias register, the Primary Error latch uses the Initial X Position latch, and the Secondary Error latch uses the Source 2 Y Step latch.

The Source 1 addressing latches can be used for addressing a linear operand for patterned (dashed and dot) vectors. The source 1 operand can either be color expanded or not. Two dimensional tiled and stippled vectors are not supported. Three operand vectors are also not supported.

The Positive and Negative Y Step latches are loaded with 14 bit 2s complement Y scan line steps. The Positive or Negative Y Offset latch is sign extended and added to the Destination Base Address latch, dependent on the Error Accumulator register's most significant bit.

The X Position register is loaded with the starting X Pixel coordinate relative to the window or pixel map. The X Position register is added to the positive access size, negative access size in pixels, 1 or negative 1 or 0 and updated in parallel with the forming of the new Destination Base Address. The X Position will increment, decrement, or remain unchanged, depending on the octant of the vector and the most significant bit of the Error Accumulator Register.

The Next Pixel Address is the latched sum of the X Position register, X Bias, and the Destination Base Address latch, as described in the Operand Addressing section above.

The error register's most significant bit also selects between the 16 bit 2s complement primary and secondary error offset latches.

The operand addressing is pipelined for enabling the calculation of the next address, check for last pixel, window clip, and virtual address comparison to occur in parallel with the addressing of the present pixel.

The pixel size will be within the address generator control register for helping the address generator control state machine determine the number of bytes which must be read and written. The X Position and Bias latches address to either the bit for monochrome or byte for 8 plane color.

Zero length vectors will draw no points. Vectors of length 1 will draw one pixel and no registers other than the Destination Base latch, X Position Register, and the vector length counter need to be loaded.

Typically, the data for flat vectors is supplied in the Pixel SLU's Foreground latch with the logical function of pass source.

Vectors drawn to main memory, DRAM, will generate longword read-modify-write accesses.

5.6.4 Window Clipping Rectangle

The address generator supports window clipping rectangles. The clipping hardware is loaded with one rectangle at a time and the primitive is executed. All graphics primitives can use the window clipping rectangle functionality to both the frame buffer and main memory.

There are 4 clipping rectangle latches x, y minimum and x, y maximum. The window clipping rectangle is loaded with the linear longword addresses for the y coordinate for the upper and lower left corner of the window, screen, or pixel map. The y latches and comparator are 28 bits each. The x values are relative to these longword addresses. The x latches and comparators are 16 bits each.

There are 2 comparators used twice per destination access which check the four edges of the rectangle. Writing will occur when the address is within the rectangle. The X comparator checks multiple pixels at at time: 16 for 8 plane and 32 for monochrome. This means that monochrome or contiguous single plane operations will be performed a longword (32 pixels) at a time, while color can operate on an octaword (16 pixels) at a time.

The drawing operation can be continued, but no writes nor memory cycles occur or the addressing can stop immediately, when the drawing leaves the rectangle. Both options exist.

6 PVAX2 LCG Address Ranges:

Address Range	Data Size:	Description
0000.0000 - 07FF.FFFF	128M bytes	Main Memory
2004.0000 - 2007.FFFF	256k bytes	System ROM
2010.0000 - 2013.FFFF	256k bytes	LCG Internal Registers
2014.0000 - 2017.FFFF	256k bytes	LCG Frame Buffer ROM
2018.0000 - 201F.FFFF	512k bytes	LCG Command FIFO
2100.0000 - 22FF.FFFF	32M bytes	LCG Frame Buffer Memory
2100.0000 - 22FF.FFFF	32M bytes	MV_DAL Option (Imaging, 3D Scan Proc,...)
2100.0000 - 217F.FFFF	8M bytes	Plane Masked and Write Buffered
2180.0000 - 21FF.FFFF	8M bytes	Non-Plane Masked and Write Buffered
2200.0000 - 227F.FFFF	8M bytes	Plane Masked and NOT Write Buffered
2280.0000 - 22FF.FFFF	8M bytes	Non-Plane Masked and NOT Write Buffered

7 PVAX2 LCG Direct Frame Buffer Access

The frame buffer can be accessed by directly addressing any location in this I/O space address range. The frame buffer memory is byte addressable. Reads will return a contiguous longword from either the monochrome or color frame buffer. The normal LCG Frame Buffer address space is mapped 4 times with all permutations of: Plane masked, Non-Plane masked, CPU Write Buffered, and CPU NOT Write Buffered.

7.1 Frame Buffer Accesses with Plane Masked Writes

All writes will use the plane mask for writing individual planes. The monochrome frame buffer should not be addressed through this address space unless the plane mask is set to FF. The plane mask has no meaning with monochrome except for performing a special effect with masking a 8 pixel pattern.

Address Mode	Address Range	Data Size (bytes)
LCG Frame Buffers	2100,0000:217F,FFFF	8,388,608
8 plane 1M Pixel	2100,0000:210F,FFFF	1,048,576
8 plane 2M Pixel	2100,0000:211F,FFFF	2,097,152
Mono 4M Pixel	2100,0000:2107,FFFF	524,288

7.2 Frame Buffer Accesses with No Plane Masked Writes

The Write Mask will not be used. The monochrome frame buffer should be accessed through this address range.

Address Mode	Address Range	Data Size (bytes)
LCG Frame Buffers	2180,0000:21FF,FFFF	8,388,608
8 plane 1M Pixel	2180,0000:218F,FFFF	1,048,576
8 plane 2M Pixel	2180,0000:219F,FFFF	2,097,152
Mono 4M Pixel	2180,0000:2187,FFFF	524,288

8 Main Memory, Video Frame Buffer, & 3D Scan Proc Addressing

This section describes the addressing mechanism for supporting the different types of memory within the PVAX2 LCG system.

8.0.0.1 DRAM & VRAM Address Pins Table

V/DRAM Type:	Row Pins	Columm Pins	RASENs
128kx8	9	8	2 decoded to 4
256kx4	9	9	2 decoded to 4
1Mx1	10	10	2 decoded to 4
4Mx1	11	11	2 decoded to 4

8.1 Memory, Frame Buffer, Imaging, 3D Scan Proc & ROM Address table

The 256kx4 Main Memory implementation uses 1Mx1s for the Parity bits. Address bits <22:21> are asserted along with the RAS and CAS addresses, as well as, creating the RASEN (bank select) bits. The RAS for the parity bits will not be be gated with the RASEN bits, but will always be accessed when ever main memory is accessed. The 256kx4 VRAMs address mode will be accessed exactly like the 256kx4 main memory for consistancy.

Memory & FB Addressing	Row Address	Column Address	RASENs Address	
256kx4 M Mem	22,20:12	21,11:03	dec<22:21>	(2,4,6,8Mb)
1Mx1 M Mem	22:13	12:03	dec<24:23>	(8,16,24,32Mb)
4Mx1 M Mem	24:14	13:03	dec<26:25>	(32,64,96,128Mb)
128kx8 1P FB(32)	18:10	09:02	dec<20:19>	(.5,1,1.5,2Mb)
128kx8 8P FB	19:11	10:03	dec<21:20>	(1,2,3,4Mb)
256kx4 8&32P FB	22,20:12	21,11:03	dec<22:21>	(2,4,6,8Mb)
3D SP	22:13	12:03	dec<24:23>	(8,16,24,32Mb)
ROM	CAS<3:0>=20,~20,17:16	11:02	MVBUS=15:12	(256kb+256kb)

8.2 Memory Access Addressing

All normal memory references come from the normal (described in the table above) address bits. This section describes where the Memory Refresh, Video Refresh, and RAMDAC LUT Loads RAS Addresses are taken from.

Memory & FB Addressing	Row Address	Column Address	RASENs Address
Memory Refresh	24:14	Don't Care	All V&DRAM Asserted
Video & LUT Loads	24:14	Don't Care	Dependant on the Memory Config MEM_VIDEO_MASK bits.

9 Video Subsystem

The Video Subsystem supports both monochrome and 8 plane color with a 2 plane cursor. All video display subsystems reside on separate modules. Currently, 3 8 plane video modules are in development: 1M pixel 8 plane module (supports both 1024x768 and 1024x864), 2M pixel 8 plane module (supports 1280x1024), and a dual 2M pixel 8 plane module (supports 1280x1024). A True Color (24 planes plus 8 overlays) video module also is in developement (supports 1280x1024). A monochrome module is possible, but not currently in development.

The video timing state machine is programmable with the values for number of active pixels and scan lines, as well as, horizontal and vertical front porch, sync pulse, and back porch timing parameters. All horizontal timing parameters must be a multiple of 4 pixels. Resolutions up to 2048x2048 can be logically accomidated. It is unclear at this time what the timing contraints will turn out to be.

The screen resolutions and video timing parameters will be contained within the Video Module's Diagnostic ROM. The appropriate Crystal must used.

9.1 8 Plane Frame Buffers

All 8 plane displays use the Brooktree BT458 RAMDAC for cost reasons. The BT458 is on average $30-$60 less than the BT459. Since cost is a priority for the Low Cost Graphics system, the LEGO Pix Maps and Window Cursor chips will NOT be used.

The 8 plane frame buffer is accessed exactly like main memory which is 64 bits wide. All main memory access sizes are supported. The frame buffer memory allows individual bit writes with the use of the plane mask which is loaded on the assertion of RAS.

9.2 Cursor

A 64x64x2 cursor sprite is multiplexed with the output of the color lookup table. For both monochrome and color, a scan line of cursor will be prefetched from main memory for each cursor active scan line. A cross hair cursor will not be supported, since the BT459 is not included in the 8 plane design.

The Cursor is stored in main memory and NOT off-screen frame buffer memory. The contiguous memory required to support a 64x64x2 cursor is 64 octawords which is 1024 bytes, and is aligned on a natual 1KB boundry. The Cursor cannot be stored in off-screen monochrome frame buffer memory, since the largest access size supported is longword which is not large enough to access a scan line of cursor with 1 access.

The Cursor Starting Address will be 1024 byte aligned. The address latch will be 25-9=16 bits. There is a 6 bit counter which is used to address the individual octawords.

The cursor hotspot is located at the upper left corner of the cursor pattern. Cursor location (X=0,Y=0) positions the cursor pattern in the upper left corner of the screen.

In order to position the cursor off the top of the screen, Bit 31 of the Y_start register must be a 1, while the first visible scan line is loaded into bits <21:16>. In order to position the cursor totally off the top or bottom of the screen, load 1s into bits <31:30>.

In order to position the Cursor off the left of the screen, bit 15 of the X_Start register must be a 1 and the appropriate code must be loaded into bits <6:0>. In order to position the cursor totally off the left side of the screen, load 1s into bits <15:14>. The value can be generated with the following VAX macro code:

```
; R0 = Cursor X position and negative
MNEGL   R0,R0           ; Amount off the left edge
DECL    R0              ; So adjustment works
XORL    #3,R0           ; Fix lower 2 bits
ADDL    #^X8004,R0      ; Fix other bits
```

9.3 Video Output Data

The diagram below shows the addressing for 2 longwords which describe 8 pixels. Notice the addressing of each byte and longword.

```
        VAX MEMORY PIXELS
|31                      00|31                      00| ect...
-------------------------------------------------------------
| xx03 | xx02 | xx01 | xx00 | xx07 | xx06 | xx05 | xx04 | ... |
-------------------------------------------------------------
```

The Frame Buffer will be wired to correspond to the diagram below.

```
        SCREEN PIXELS

    top left pixel as viewed on screen
       /
      /
|00  /                   31|00                      31| ect...
-------------------------------------------------------------
| xx00 | xx01 | xx02 | xx03 | xx04 | xx05 | xx06 | xx07 | ... |
-------------------------------------------------------------
```

9.4 8 Plane Look-Up Table (LUT) Data

The LUT will be loaded on every vertical retrace. The Console Enable bit will determine whether the Console address will be used for the LUT data or the loadable Server Address. When the CTL LUT address is loaded, the next vertical retrace will use this address for loading the Cursor Overlay registers and other RAMDAC control registers.

The 8 plane RAMDAC is loaded from the Frame Buffer offscreen memory. The format in off-screen VRAM maps directly to the DECwindows LUT format minus the R,G, and B valid masks and the Pixel Value Address and size. The DECwindows color values are 16 bits of precision. The most significant 8 bits will be stored within the Brooktree 458 RAMDAC LUT. The format will be:

The following is an diagram of the LUT organization for the server's color palette. This format packs the data less efficiently since it does not take advantage of BT458's internal autoincrement counter. However, maintaining quadword alignment makes it easier for the server to locate and modify an arbitary color map entry.

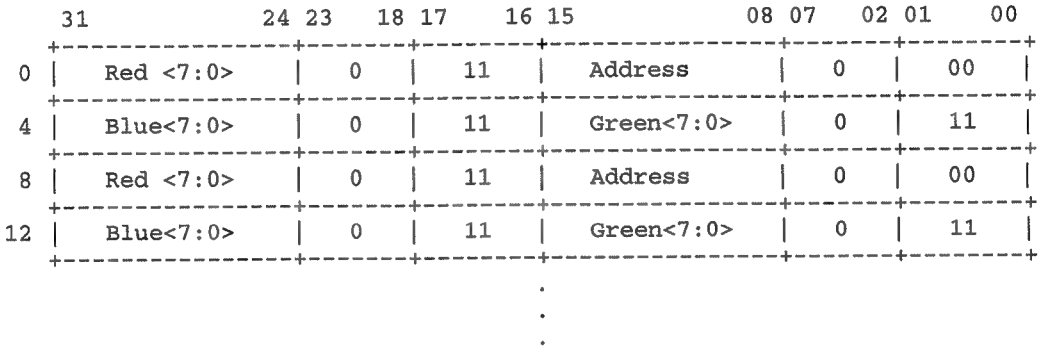


Table 2: Possible Values of LUT Data Bits <01:00>

Name	Value	Description
LUT_ADRS_REG	00	Write to LUT address register
LUT_COLOR_AUTOINC	01	Write color to LUT and autoincrement address

9.5 8 Plane Look-Up Table Load Mechanism

The LUT data can take up either 512 or 1024 longwords of off-screen memory space which is 1 video shift register load. The Server LUT Data address will be contained in the LUT address latch. The address will be aligned to the shift register, which is modulo 2048 or 4096. Including the console, 4 2048 byte LUT data spaces are required which is 8,192 bytes. The LUT address latch is 9 bits in length.

The Console LUT Data is always the least significant address within the frame buffer. The LUT Console-select bit directs the video subsystem to use the Console LUT data whenever told to load the BT458 LUT. The LUT Load Bit signals the video subsystem to reload the BT458 LUT.

9.6 Console LUT and Display Mechanism

To switch to console:

- Assert console-select bit.
- Server will continue to load Server LUT latch LUT will be reloaded from address 0 because console-select bit is on.

To switch out of console:

- Clear console-select bit.
- Server LUT will be reloaded on the next Vertical Retrace.

To load new server LUT:

- Load LUT data - any number of LUT buffers allowed.
- Load LUT Address Latch.

9.7 Video Shift Register Loads

The following table shows the number of times the video shift register must be loaded in both scan lines and pixels. The next column shows the number of milliseconds used with a 30ns SOC and (except for the 800x600) 66 Hz refresh rate for video shift register loads and the percentage is the percent of the total memory bandwidth taken for video refresh. If the refresh rate is 72 Hz, then this time will increase by 72/66 = 17%.

Resolution and VRAM Type (Refresh Rate)	8 Plane Video				1 Plane Video			
	S.Ls.	Pixels	Millisecs	%	S.Ls.	Pixels	Millisecs	%
800x 600(128kx8) (60 Hz)	1.28	1024	6.7536	6.8	5.12	4096	1.6920	1.7
1024x 768(128kx8) (66 Hz)	2.00	2048	6.0826	6.1	8.00	8192	1.5207	1.5
1280x1024(128kx8)	0.80	1024	20.2752	20.3	3.20	4096	5.0688	5.6
1280x1024(256kx4) (66 Hz)	1.60	2048	10.1376	10.1*	N.A.			
1536x1152(128kx8)	0.67	1024	27.0547	27.1	2.66	4096	6.7637	6.8
1536x1152(256kx4) (66 Hz)	1.33	2048	13.5274	13.5*	N.A.			

The 1024x768 video can use the full shift register loads, since an even number of scan lines can fit into the shift registers. All other resolutions must use the split shift register loads which automatically doubles the number of loads. The 1280x1024 and 1536x1152 8 plane frame buffer uses the 256kx4 VRAMs which doubles the amount of video shift register available. This is also programmable, where a 2 bit code defines how often to reload the the video shift register.

Bit Code:	Shift Register Load
0	1024 pixels
1	2048 pixels
2	4096 pixels
3	8192 pixels

The color frame buffers access 8 pixels at a time while the monochrome frame buffer accesses 32 pixels. The S-Chip supports 32 bit monochrome frame buffers.

The following table shows the percentage of total memory bandwidth that the supported resolutions take for video refresh at all refresh rates.

	8 Plane Video			1 Plane Video		
Resolution:	60Hz	66Hz	72Hz	60Hz	66Hz	72Hz
800 x 600	6.8	7.5	8.2	1.7	1.9	2.0
1024 x 768	5.5	6.1	6.7	1.4	1.5	1.7
1280 x 1024	9.2	10.1	11.0	4.6	5.1	5.5
1536 x 1152	12.3	13.5	14.7	6.2	6.8	7.9

9.8 Console Active Video Data

For a 100 dpi monitor, the Console Video Data requires a minimum of 6 lines of text with the text height of 21 scan lines which equals 126 scan lines. The console video data cannot end in the with a partial video shift register. The 1280x1024 color and monochrome monitors require the Console Video Data to be 128 scan lines for ending with an empty video shift register which results in:

```
128 x 1280 pixels      = 163,840 pixels
8 plane 168,960/1024   = 160 split shift register loads
Mono    168,960/2048   =  40 split shift register loads

128 x 1024 pixels      = 131,072 pixels
8 plane 131,072/2048   =  64 full shift register loads
Mono    131,072/4096   =  16 full shift register loads

128 x  800 pixels      = 102,400 pixels
8 plane 102,400/1024   = 100 split shift register loads
Mono    102,400/2048   =  25 split shift register loads

1.5x128 x 1536 pixels = 294,912 pixels (~150 dpi vs 100 dpi)
8 plane 294,912/2048   = 144 split shift register loads
Mono    294,912/4096   =  36 split shift register loads - 32 bits
```

9.9 Frame Buffer Data Allocation

The following tables shows the data which is stored within each frame buffer configuration in pixels.

9.9.1 Frame Buffer Data Allocations

The first table are the known monitors. The second table are monitors which have been requested by Marketing. The 800x600 monitor is meant to be the commodity monitor used with PCs and/or MAC-2s. The lowest cost color system would not include a monitor. The customer could keep the monitor from their obsolete PC or MAC-2. The 1536x1152 is a higher resolution monochrome, approximately 150 dpi (dots per inch).

Supported Frame Buffers:

Frame Buffer	Monochrome (Pixels)		8 Planes (Pixels)	
VRAM Data	1024 x 768	1280 x 1024	1024 x 768	1280 x 1024
Total Pixels	4,194,304	4,194,304	1,048,576	2,097,152
Active Video	786,432	1,310,720	786,432	1,310,720
Console Video	131,072	163,840	131,072	163,840
LUT Data	0	0	8,192	8,192
Off Screen Left	3,276,800	2,719,744	122,880	614,400

Proposed Frame Buffer Support:

Frame Buffer	Monochrome (Pixels)		8 Planes (Pixels)	
VRAM Data	1536x1152	800 x 600	1536x1152	800 x 600
Total Pixels	4,194,304	4,194,304	2,097,152	1,048,576
Active Video	1,769,472	481,280	1,769,472	481,280
Console Video	294,912	102,400	294,912	102,400
LUT Data	0	0	8,192	8,192
Off Screen Left	2,129,920	3,610,624	24,576	456,704

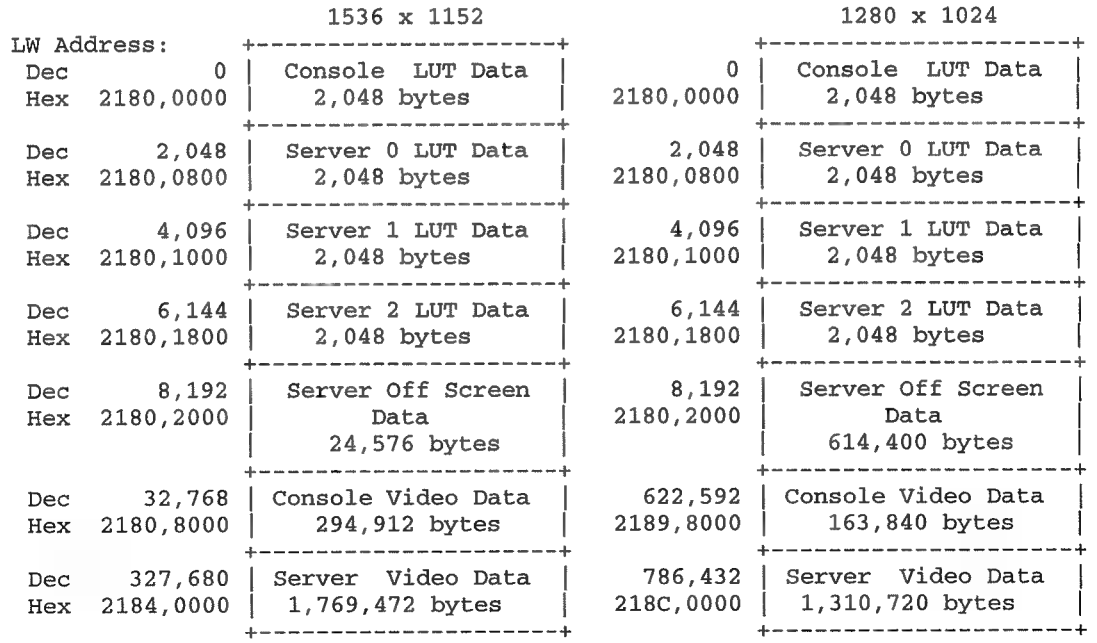
9.9.2 Color Frame Buffer Data Map

The addresses below are the suggested starting addresses for each section. With the exception of the Console LUT table, which is fixed at physical address 2180.0000, all regions are fully programable (subject to alignment restrictions). All monochrome resolutions use 262,144 bytes or 2 million pixels. There are two veriations for 8 plane video: 1 and 2 million bytes/pixels.

8 Plane - 1 Million Pixel (1024x768 & 800x600) Color Frame Buffer Data Map

LW Address:	1024 x 768	LW Address:	800 x 600
Dec 0 / Hex 2180,0000	Console LUT Data 2,048 bytes	Dec 0 / Hex 2180,0000	Console LUT Data 2,048 bytes
Dec 2,048 / Hex 2180,0800	Server 0 LUT Data 2,048 bytes	Dec 2,048 / Hex 2180,0800	Server 0 LUT Data 2,048 bytes
Dec 4,096 / Hex 2180,1000	Server 1 LUT Data 2,048 bytes	Dec 4,096 / Hex 2180,1000	Server 1 LUT Data 2,048 bytes
Dec 6,144 / Hex 2180,1800	Server 2 LUT Data 2,048 bytes	Dec 6,144 / Hex 2180,1800	Server 2 LUT Data 2,048 bytes
Dec 8,192 / Hex 2180,2000	Server Off Screen Data 122,880 bytes	Dec 8,192 / Hex 2180,2000	Server Off Screen Data 456,704 bytes
Dec 131,072 / Hex 2182,0000	Console Video Data 131,072 bytes	Dec 464,896 / Hex 2187,1800	Console Video Data 102,400 bytes
Dec 262,144 / Hex 2184,0000	Server Video Data 786,432 bytes	Dec 567,296 / Hex 2188,A800	Server Video Data 481,280 bytes

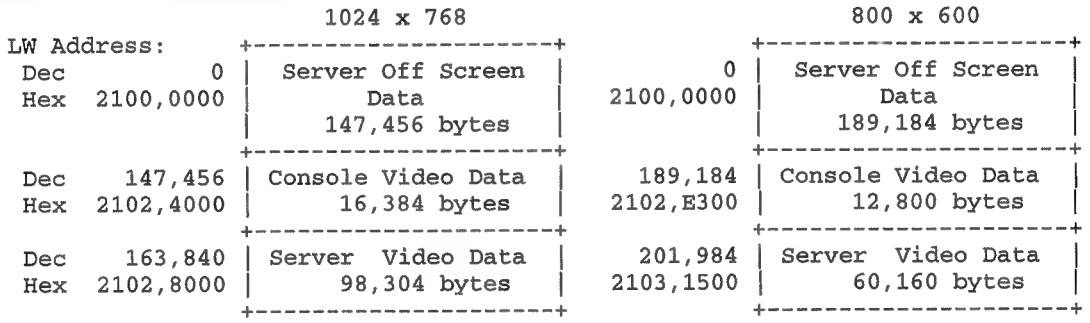
8 Plane - 2 Million Pixel (1280x1024 & 1536x1152) Color Frame Buffer Data Map



The 1536x1152 fits within the current 2 million pixel frame buffer, but the off-screen memory is very limited. This may not be a problem because of the virtual drawing capabilities of LCG.

9.9.3 Monochrome Frame Buffer Data Map

Monochrome 1024x768 & 800x600 Frame Buffer Data Map



Monochrome 1280x1024 & 1536x1152 Frame Buffer Data Map

Both displays fit within the current 2 Million pixel monochrome display. The 1536x1152 resolution is short on off-screen memory, but with virtual drawing, this may not be a problem.

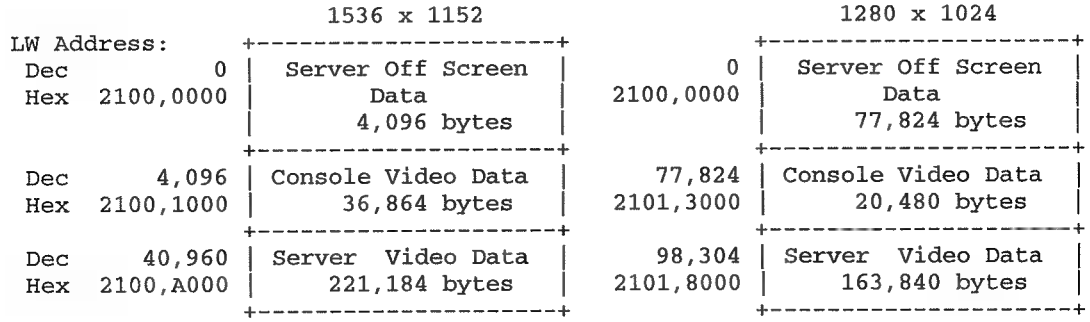


Figure 11: Video State Machine Block Diagram

Figure 12: Low Resolution 8 Plane Video Module Block Diagram

Figure 13: High Resolution 8 Plane Video Module Block Diagram

Figure 14: Dual High Resolution 8 Plane Video Module Block Diagram

Figure 15: High Resolution True Color Video Module Block Diagram

9.10 Possible VRAM Configurations

The current strategy is to use the 128kx8 1M bit VRAMs for the low resolution 8 plane frame buffer and the 256kx4 1M bit VRAMs for all other configurations. The frame buffers are controlled identically as Main Memory except for the special functions, such as, write per bit masks and video shift register loads.

The 64kx4 VRAMs do not have the split shift register functionality. Memory Engineering has warned Engineering to stop designing this part into products for both cost considerations, as well as, this part will become unavailable during FY92.

If a monochrome frame buffer is built, it would use 4 128kx8 VRAMs. This simplifies the Graphics Controller significantly and makes the monochrome data manipulation identical to the color frame buffer and main memory. Accesses to the monochrome frame buffer are limited to longword.

9.11 Video Modules Transfer Cost

The Video Modules are designed for minimal cost. The Graphics Controller is free. The Minimal number of VRAMs & Video Out circuitry is the only cost.

9.12 PVAX2 LCG Monitor Timing

This section details the monitor timing supported by the Video Controller. All timing parameters scale with the refresh rate for each resolution. The timing parameters apply to both monochrome and color. The hexidecimal numbers with (*) next to them are the numbers which reside in the Video Diagnostic ROM and will be loaded into the video state machine. The following timing parameters have been supplied by Raggy Arumugham and Brian McLane.

During Vertical Sync Pulse, Horizontal Sync Front Porch, HFRONTPORCH, will be used for the serrated pulses (Sync deassertion). The following equations define the Blank and Sync signal pins:

```
BLANK   := HBLANK OR VBLANK
SYNC    := [VSYNC AND (NOT HFRONTPORCH)] OR [(NOT VSYNC) AND HSYNC]
```

These signals will be asserted low.

9.12.1 Monitor Timing Table - VGA+ = 800 x 600 @ 60 Hz

800 x 600 @ 60Hz		
PIXEL CLOCK	44.9000 MHz = 22.272 ns	
NIBBLE CLOCK	11.2249 MHz = 89.088 ns	
HORIZONTAL FREQ	38.1803 kHz = 26.192 us	
VERTICAL FREQ	59.2861 Hz = 16.867 [illegible]	

HORIZONTAL SCAN	MICROSECONDS 60Hz	PIXELS	NUMBER OF NIBBLES	
Entire Line	26.1915	1176	294 = 126 (Hex)	
Active (Visible) Line	17.8174	800	200 = C8 (Hex)	*
Blanking Interval	8.3742	376	94 = 5E (Hex)	
Sync Front Porch	1.3363	60	15 = F (Hex)	*
Sync Pulse	4.4543	200	50 = 32 (Hex)	*
Sync Back Porch	2.5835	116	29 = 1D (Hex)	*

VERTICAL SCAN	MILLISECONDS 60Hz	NUMBER OF LINES	
Entire Frame	16.8673	644 = 284 (Hex)	
Visible Raster	15.7149	600 = 258 (Hex)	*
Blanking	1.1524	44 = 2C (Hex)	
Sync Front Porch	0.3143	12 = C (Hex)	*
Sync Pulse	0.3667	14 = E (Hex)	*
Sync Back Porch	0.4714	18 = 12 (Hex)	*

9.12.2 Monitor Timing Table - 1024 x 768 @ 60 Hz

1024 x 768 @ 60Hz		
PIXEL CLOCK	64.0000 MHz = 15.625 ns	
NIBBLE CLOCK	16.0000 MHz = 62.500 ns	
HORIZONTAL FREQ	48.7800 kHz = 20.500 us	
VERTICAL FREQ	60.0000 Hz = 16.667 [illegible]	

HORIZONTAL SCAN	MICROSECONDS 60Hz	PIXELS	NUMBER OF NIBBLES	
Entire Line	20.5000	1312	328 = 148 (Hex)	
Active (Visible) Line	16.0000	1024	256 = 100 (Hex)	*
Blanking Interval	4.5000	288	72 = 48 (Hex)	
Sync Front Porch	1.0000	64	16 = 10 (Hex)	*
Sync Pulse	1.5000	96	24 = 18 (Hex)	*
Sync Back Porch	2.0000	128	32 = 20 (Hex)	*

VERTICAL SCAN	MILLISECONDS 60Hz	NUMBER OF LINES	
Entire Frame	16.6667	813 = 32D (Hex)	
Visible Raster	15.7440	768 = 300 (Hex)	*
Blanking	0.9225	45 = 2D (Hex)	
Sync Front Porch	0.0615	3 = 3 (Hex)	*
Sync Pulse	0.0615	3 = 3 (Hex)	*
Sync Back Porch	0.7995	39 = 27 (Hex)	*

9.12.3 Monitor Timing Table - 1024 x 768 @ 66 Hz & 72 Hz

	1024 x 768 @ 66Hz	1024 x 768 @ 72Hz
PIXEL CLOCK	66.2500 MHz = 15.094 ns	72.4040 MHz = 13.811 ns
NIBBLE CLOCK	16.5629 MHz = 60.376 ns	18.1015 MHz = 55.244 ns
HORIZONTAL FREQ	52.7468 kHz = 18.958 us	57.6465 kHz = 17.239 us
VERTICAL FREQ	66.0987 Hz = 15.129 ms	72.2387 Hz = 13.843 ms

HORIZONTAL SCAN	MICROSECONDS 66Hz	72Hz	NUMBER OF PIXELS	NIBBLES	
Entire Line	18.9585	17.3471	1256	314 = 13A (Hex)	
Active (Visible) Line	15.4566	14.1429	1024	256 = 100 (Hex)	*
Blanking Interval	3.5019	3.2042	232	58 = 3A (Hex)	
Sync Front Porch	0.2415	0.2210	16	4 = 4 (Hex)	*
Sync Pulse	1.3283	1.2154	88	22 = 16 (Hex)	*
Sync Back Porch	1.9321	1.7679	128	32 = 20 (Hex)	*

VERTICAL SCAN	MILLISECONDS 66Hz	72Hz	NUMBER OF LINES	
Entire Frame	15.1289	13.8430	798 = 31E (Hex)	
Visible Raster	14.5601	13.3226	768 = 300 (Hex)	*
Blanking	0.5688	0.5204	30 = 1E (Hex)	
Sync Front Porch	0.0569	0.0520	3 = 3 (Hex)	*
Sync Pulse	0.0569	0.0520	3 = 3 (Hex)	*
Sync Back Porch	0.4550	0.4163	24 = 18 (Hex)	*

9.12.4 Monitor Timing Table - 1280 x 1024 @ 66 Hz & 72 Hz

	1280 x 1024 @ 66Hz	1280 x 1024 @ 72Hz
PIXEL CLOCK	119.8430 MHz = 8.3443ns	130.8080 MHz = 7.6448ns
NIBBLE CLOCK	29.9606 MHz = 33.3772ns	32.7020 MHz = 30.5792ns
HORIZONTAL FREQ	70.6617 kHz = 14.1519us	77.1272 kHz = 12.9655us
VERTICAL FREQ	66.4739 Hz = 15.0435ms	72.5562 Hz = 13.7824ms

HORIZONTAL SCAN	MICROSECONDS 66Hz	72Hz	NUMBER OF PIXELS	NIBBLES	
Entire Line	14.1519	12.9655	1696	424 = 1A8 (Hex)	
Active (Visible) Line	10.6806	9.7853	1280	320 = 140 (Hex)	*
Blanking Interval	3.4712	3.1802	416	104 = 68 (Hex)	
Sync Front Porch	0.2670	0.2446	32	8 = 8 (Hex)	*
Sync Pulse	1.3351	1.2232	160	40 = 28 (Hex)	*
Sync Back Porch	1.8691	1.7124	224	56 = 38 (Hex)	*

VERTICAL SCAN	MILLISECONDS 66Hz	72Hz	NUMBER OF LINES	
Entire Frame	15.0435	13.7824	1063 = 427 (Hex)	
Visible Raster	14.4916	13.2767	1024 = 400 (Hex)	*
Blanking	0.5519	0.5056	39 = 27 (Hex)	
Sync Front Porch	0.0425	0.0389	3 = 3 (Hex)	*
Sync Pulse	0.0425	0.0389	3 = 3 (Hex)	*
Sync Back Porch	0.4670	0.4279	33 = 21 (Hex)	*

9.12.5 Monitor Timing Table - 1536 x 1152 @ 66 Hz & 72 Hz

	1536 x 1152 @ 66Hz	1536 x 1152 @ 72Hz
PIXEL CLOCK	146.3827 MHz = 6.8314ns	160.0000 MHz = 6.2500ns
NIBBLE CLOCK	36.5957 MHz = 27.3256ns	40.0000 MHz = 25.0000ns
HORIZONTAL FREQ	78.8700 kHz = 12.6791us	86.2069 kHz = 11.6000us
VERTICAL FREQ	66.0000 Hz = 15.1515ms	72.1397 Hz = 13.8620ms

HORIZONTAL SCAN	MICROSECONDS 66Hz	MICROSECONDS 72Hz	NUMBER OF PIXELS	NUMBER OF NIBBLES	
Entire Line	12.6791	11.6000	1856	464 = 1D0 (Hex)	
Active (Visible) Line	10.4930	9.6000	1536	384 = 180 (Hex)	*
Blanking Interval	2.1860	2.0000	320	80 = 50 (Hex)	
Sync Front Porch	0.5465	0.5000	80	20 = 14 (Hex)	*
Sync Pulse	1.0930	1.0000	160	40 = 28 (Hex)	*
Sync Back Porch	1.5465	0.5000	80	20 = 14 (Hex)	*

VERTICAL SCAN	MILLISECONDS 66Hz	MILLISECONDS 72Hz	NUMBER OF LINES	
Entire Frame	15.1515	13.8620	1195 = 4AB (Hex)	
Visible Raster	14.6063	13.3632	1152 = 480 (Hex)	*
Blanking	0.5452	0.4988	43 = 2B (Hex)	
Sync Front Porch	0.0127	0.0116	1 = 1 (Hex)	*
Sync Pulse	0.0254	0.0232	2 = 2 (Hex)	*
Sync Back Porch	0.5198	0.4756	41 = 29 (Hex)	*

9.13 Video State Machine Timing

This section describes the inner workings of the Video State Machine's timing with pipeline delays accounted for.

9.13.1 Video Timing State Machine

The Video Timing State Machine is designed to be programmable for supporting multiple video timing parameters for supporting a flexible video system.

9.13.1.1 Horizontal State Timing

The 9 bit synchronous Horizontal Pixel Counter is clocked off the Nibble Clock's rising edge (NIB_CLK H) which is the pixel clock divided by 4. The Horizontal Pixel Counter is synchronously cleared by both Reset and the HCLR_CNT H signal. The HCLR_CNT H signal is generated by the 2 bit synchronous Horizontal State Counter and 9 bit Horizontal Pixel Comparitor. The Horizontal State Counter selects the appropriate horizontal timing parameter from the Horizontal Timing latch. The HCLR_CNT H signal is asserted 1 cycle after the comparison is true and the Horizontal Pixel Counter is cleared 1 cycle later. The rising edge of HCLR_CNT H also increments the Horizontal State Counter.

The Horizontal Pixel Counter increments 1 value past its limit, and is cleared on the next NIB_CLK H clock cycle. The Horizontal Pixel Counter takes (horizontal timing parameter + 2) cycles between HCLR_CNT H signals. A timing parameter of 0 DOES NOT WORK. A timing parameter of 1 is the smallest allowable value and results in a 3 cycle time period. The HCLR_CNT H signal is asserted for 1 NIB_CLK H clock cycle.

The NEW_SL H signal is generated by Horizontal State Counter equaling 0 which is the state for Horizontal Front Porch. This signal is generated 1 cycle after the Horizontal State Counter equals 0 and is deasserted 1 cycle after the Horizontal State Counter equals 1, Horizontal Sync Pulse. NEW_SL H is asserted high for the duration of Horizontal Front Porch.

H_PIX<10:02> H Sequence:

Hor Front Porch	Hor Sync Pulse	Hor Back Porch	Hor Active Video
0 1 2 . . Lim Lim+1	0 1 . . Lim Lim+1	0 1 . . Lim Lim+1	0 1 . . Lim Lim+1
H_STATE<1:0>=0	H_STATE<1:0>=1	H_STATE<1:0>=2	H_STATE<1:0>=3

9.13.1.2 Vertical State Timing

The 11 bit synchronous Vertical Scan Line Counter is clocked on the rising edge of the NEW_SL H signal which is generated by the Horizontal Timing above. The Vertical Scan Line Counter is cleared by both Reset and the VCLR_CNT H signal. The VCLR_CNT H signal is generated by the 2 bit synchronous Vertical State Counter and 11 bit Vertical Scan Line Comparitor. The Vertical State Counter selects the appropriate vertical timing parameter from the Vertical Timing latch, just as the Horizontal state machine described above. The VCLR_CNT H signal is asserted 1 cycle after the comparison is true. The rising edge of VCLR_CNT H also increments the Vertical State Counter.

The Vertical Scan Line Counter starts incrementing 1 value past its limit and is cleared 1 NIB_CLK H clock cycle later and remains cleared until the next NEW_SL H rising edge. The Vertical Scan Line Counter takes (vertical timing parameter + 1) cycles between VCLR_CNT H signals. A timing parameter of 0 DOES NOT WORK. A timing parameter of 1 is the smallest allowable value and results in a 2 cycle time period. The VCLR_CNT H signal is asserted for 1 clock cycle (NIB_CLK H).

V_PIX<10:00> H Sequence:

Ver Front Porch	Ver Sync Pulse	Ver Back Porch	Ver Active Video
0 1 2 . . . Limit	0 1 . . . Limit	0 1 . . . Limit	0 1 . . . Limit
V_STATE<1:0>=0	V_STATE<1:0>=1	V_STATE<1:0>=2	V_STATE<1:0>=3

9.13.2 Memory Requests for Video & Memory Refresh

This section describes the timing of Video Refresh, LUT Video Shift Register Loads, Cursor Data Octaword Reads, and Memory Refresh memory accesses. The Cursor Base Address Latch, Load LUT Address Latch, and Video Refresh Address Counter are loaded during Horizontal Front Porch of the first scan line of Vertical Front Porch from the CPU loadable latches. The VB_CLR H signal is synchronized to PCLK3 H and produces a 1 cycle load pulse (VT_LD H) which loads the Cursor Base Address, Load LUT Address latches and the Video Refresh Address Counter directly.

The following memory access timing sections are listed in priority order.

9.13.2.1 Video Refresh Memory Access Timing

This memory access is masked when BLANK EN L (VID_CFG1 H = 0) is asserted.

The first Video Refresh Memory Access for the new frame is requested during the Horizontal Front Porch of the first scan line and is initiated by the VREF1 L signal generated by the Video Timing State Machine. All other Video Refresh Memory accesses are the result of the Video Refresh Counter.

The Video Refresh Counter is clocked by the rising edge of NIB_CLK H and allowed to count by VCNT_EN H (V_STATE<1:0> = 3 and H_STATE<1:0> = 3). The number of pixels between video refreshes is programmable with 4 values and the number is a multiple of 4.

Binary Code	Number of Pixels
0	1024 pixels
1	2048 pixels
2	4096 pixels
3	8192 pixels

The type of video shift register load is also programmable through the Video Configuation register. The 2 choices are split (1/2) loads and full loads. If split loads are selected, then the top-of-frame shift register load is a full load, but all remaining loads are split. The double buffered video shift register must be primed with both halves being full.

9.13.2.2 Cursor Data Memory Access Timing

This memory access is masked when CURSOR EN H (VID_CFG2 H = 0) is deasserted.

The octaword reads are requested at the end of Horizontal Front Porch of Scan Lines where the CUR_YHIT L signal is asserted. This gives the Horizontal Sync and Horizontal Back Porch accumulative time to fetch the Cursor Data for displaying during the active video of the current scan line.

9.13.2.3 LUT Video Shift Register Load Timing

This memory access is masked when LUT_LOAD_EN H (VID_CFG3 H = 0) is NOT asserted.

The LUT Video Shift Register Load is requested at the beginning of Vertical Front Porch from the VT_LD H signal.

9.13.2.4 Memory Refresh Access Timing

The Memory Refresh memory request is controlled identically to the Video Refresh Memory Accesses, but is delayed by 1 NIB_CLK H cycle. The delay is to to make absolutely sure that the Video Refresh is serviced before the Memory Refresh. The Memory Refresh Counter has its own select bits during active video. The granularity is finer. The number of pixels between video refreshes is programmable with 4 values and the number is a multiple of 4. During Vertical retrace, a memory refresh is generated on every scan line.

Binary Code	Number of Pixels
0	64 x 4 = 256 pixels
1	128 x 4 = 512 pixels
2	256 x 4 = 1024 pixels
3	512 x 4 = 2048 pixels

9.13.3 Video Signals Timing

This section describes the Video Timing signals (VBLANK L, V_SYNC L, and VIDSHF L), as well as, the Video Output Control signals (VMXSEL0 H, VID_BTEN L, and VIDMSEL H). These signals are in respect to the the Video State Machine timing with all pipeline stages described. All signals are clocked on the rising edge of NIB_CLK H.

9.13.3.1 VBLANK L Timing

VBLANK L is pipelined 6 cycles from H_STATE<literal(<1:0>)> = 3 with V_STATE<literal(<1:0>) = 3.

9.13.3.2 VMXSEL0 H Timing

The VMXSEL0 H signal is H_PIX02 H pipelined 1 cylce during active video (H_STATE<1:0> H equals 3 and V_STATE<1:0> H equals 3). During non-active video, this signal is LDCNT3 H pipelined 1 cycle. LDCNT<3:0> H are generated by the Brooktree RAMDAC load logic.

9.13.3.3 V_SYNC L Timing

V_SYNC L is asserted (0) pipelined 2 cycles after the following 2 states:

V_SYNC L	H_STATE<1:0> H	V_STATE<1:0> H
0	0 1	NOT (0 1)
0	NOT (0 1)	0 1

This table asserts V_SYNC L during the required time for non-Vertical Sync, as well as, generate the serated pulses for Vertical Sync.

9.13.3.4 VIDSHF L Timing

The VIDSHF L signal is asserted during both active video (H_STATE<1:0> H equals 3 and V_STATE<1:0> H equals 3) pipelined 3 cycle.

VIDSHF L is also asserted during the Brooktree RAMDAC LUT load when LDCNT<3:0> H equals F, piplelined 1 cycle.

9.13.3.5 VID_LDMSEL H Timing

The VID_LDMSEL H signal controls the Brooktree LUT load multiplexer. It is LDCNT1 H pipelined by 1 cycle.

9.13.3.6 VID_BTEN L Timing

The VID_BTEN L signal is the Brooktree LUT Enable signal. The LUT is loaded during both Vertical Sync and Vertical Back Porch. The Video Shift register load for the LUT load occurs during Vertical Front Porch. The LUT load logic loads the LUT with 1024 entries which are 256 addresses, red, green, and blue entries. Each load is 4 NIB_CLK H cycles for a total of 4096 NIB_CLK H cycles in total. The VID_BTEN L signal is LDCNT1 H pipelined 2 cycles.

9.13.4 Cursor Timing

This section describes the Cursor and timing relative to the video timing described above.

A scan line of cursor data is loaded an octaword (64x2=128 bits) at a time during horizontal retrace for display on the next active video time period.

9.13.4.1 Cursor Y Position Timing

The Cursor Scan Line Comparison samples the CUR_YCNT<10:00> H on the rising edge of NIB_CLK H 1 cycle after V_STATE<1:0> H goes to 0, Vertical Front Porch. The comparison is between the Cursor Y Position latch and CUR_YCNT<10:00> H, which is the output of the Vertical Scan Line counter. The CUR_YHIT L signal is asserted 1 cycle (NEW_SL H rising edge) after the comparison is true, BUT ONLY during Vertical Active Video. All comparisons which are true during either Vertical Front Porch, Sync, or Back Porch do NOT perform memory access.

The 6 bit counter which keeps track of the number of scan lines is incremented on the rising edge of NEW_SL H which occurs just before V_STATE<1:0> H changes state. CUR_YHIT L remains asserted for 64 scan lines (NEW_SL H rising edges). CUR_YHIT L is deasserted during Vertical Retrace.

9.13.4.2 Cursor X Position Timing

The X Position Comparitor Match register is clocked on the rising edge of NIB_CLK H. The comparison is between the Cursor X Position latch and CUR_XCNT<10:02> H, which is the output of the Cursor Pixel counter. The CUR_XHIT L signal is asserted 1 cycle after the comparison is true, BUT ONLY during Vertical AND Horizontal Active Video. CUR_YHIT L must be asserted, as well as, H_STATE<1:0> H must equal 3 (Active Video) for CUR_XHIT L to become asserted. CUR_XHIT L remains asserted for 16 clock cycles or 64 pixels.

9.13.4.3 Cursor Data Output Timing

The Cursor Data is enabled and clocked into the Video Cursor Register during the cycle that CUR_XHIT L becomes asserted. The output of the Cursor Output Register is masked by both CUR_XHIT L and CURSOR EN H (VID_CFG2 H deassertion). The Cursor Data (for 8 plane) is present at the S-Chip pin outputs 6 cycles after H_STATE<literal(<1:0>)> = 3 and V_STATE<literal(<1:0>)> = 3 for positioning on the 1st active pixel of a scan line and 2 cycles after CUR_XHIT L is asserted for all other positions.

10 PVAX2/PMariah LCG Pin and Internal Signal Definitions

This section describes pins required for the Memory System and Frame Buffers. The interface to the Memory, Graphics, and Video sub-systems from the external interfaces (SOC, SCSI, and NI) are also described here. The last section describes the internal LCG data and control signals.

10.1 Memory, Frame Buffer, Video Output, System ROM - 88 Pins

The following sections describe the S-Chip Pins which interface to the LCG subsystem.

10.1.1 Memory Address & Control Pins - 27 Pins

The following pins are the address and control signals for accessing main & video memory.

MVBUS<3:0> H is used for the main & video memory quadword bank select (RAS), the most significant 4 bits of the CAS byte mask, and system ROM address<13:10>. There are 2 separate latch enables for RAS (74ACT373 - only 4 bits needed) and CAS (74ACT373 - all 8 bits required). There are 3 separate timing signals for Main Memory RAS (4 NANDs), Video Memory RAS (2 NANDs), and CAS (8 NANDs).

LCGDSF L and LCG_DT L are required for accessing the video RAMs, VRAM, as memory, loading the plane mask, and loading the video shift register.

```
LCG Signal Name  TYPE  Description
===============  ====  ===========
 MADR<10:00> H   BT4    Main & Video Memory Adrs & ROM Adrs<11:02>
 BWRT L          BT4    Write Enable Signal - Main & Video Memory
 MVBUS<3:0> H    BT4RP  Quadword Bank (RAS) Selects: Main (MRBUS<3:0> H)
                         & Video (VRBUS<1:0> H) Memory, M_CAS<7:4> H,
                         & Sys ROM Adrs<15:12>
 MCBUS<3:0> H    BT4RP  CAS Byte Mask & M_CAS<3:0> H = ROM Adrs<17:16>
                         M_CAS<3> = SYS ROM EN, M_CAS<2> = LCG ROM EN
 MVRLEN H        BT4RP  RAS<3:0> H Latch Enable - Main & Video Memory
                         - 1 74ACT373 (only need 4 bits)
 CASLEN H        BT4RP  M_CAS<7:4> H Latch Enable
                         - 1 74ACT373
 MRAS H          BT8    RAS Timing Signal for Main Memory     - 4 NANDs
 VRAS H          BT8    RAS Timing Signal for Video Memory    - 1 NAND
                         Only 2 NANDs required for 8 plane 2M pixel
 MCAS H          BT4RP  CAS Timing Signal: Main and Video Mem - 8 NANDs
 LCGDSF L        BT4RP  VRAM Special Function (Split Shift Reg Load)
 LCG_DT L        BT4RP  VRAM Output Enable

TOTAL Signals = 27 pins
```

10.1.2 Main & Video Memory Data Interface Pins - 45 Pins

The following pins are the Memory Data signals required for accessing main memory. Main Memory and 8 plane Video Memory are 64 bits wide, while the data interconnect, MV_DAL<31:00> H, is only 32. The Monochrome Video Memory is only 16 bits wide. There are 2 sets of 32 bit latched transceivers between the MV_DAL and the Main Memory.

The Frame Buffer (Video) Output Enables, VBAOE<1:0> L and VABOE L, are separate from Main Memory, MBAOE<1:0> L and MABOE L. The main memory and 8 plane latched transceivers are controlled almost identically. Since the monochrome frame buffer is only 16 bits wide, it requires that the VRAM and MV_DAL be swapped on the latched transceivers.

LCG Signal Name	TYPE	Description	
MV_DAT<31:00> H	BD4CRP	Memory Data Interconnect	
M_PAR<3:0> H	BD4CRP	Memory Parity Bits	
LEN_AB<1:0> L	BT4RP	MV_DAL to DRAM Latch Enables	- Mem & 8 plane
LEN_BA L	BT4RP	DRAM to MV_DAL Latch Enable	- Mem & 8 plane
MBAOE<1:0> L	BT8	DRAM Data to MV_DAL Output Enables	- Memory
MABOE L	BT4RP	MV_DAL to DRAM Data Output Enable	- Memory
VBAOE<1:0> L	BT4RP	VRAM Data to MV_DAL Output Enables	- 8 plane
VABOE L	BT4RP	MV_DAL to VRAM Data Output Enable	- 8 plane

TOTAL Signals = 45 pins

10.1.3 ROM Enable - 1 Pin

The address uses the following pins for its address:

LCG Signal Name	Description
M_CAS<3> H	Sys ROM Enable L
M_CAS<2> H	LCG ROM Enable L
M_CAS<1:0> H	ROM Adrs<17:16>
MVBUS<3:0> H	ROM Adrs<15:12>
MEM_ADD<9:0> H	ROM Adrs<11:02>

The signals above are used for addressing main memory and do not add to the total pin count. The data bits are on the MV_DAL<31:00> H. The signal below is the only added pin for accessing both the System and LCG ROM. The ROMOE H signal is used for all ROMs on the MV_DAL.

LCG Signal Name	TYPE	Description
ROMOE H	BT4RP	ROM Enable

TOTAL Signals = 1 pins

10.1.4 Frame Buffer & Video Output Control Pins - 7/15 Pins

The first 4 pins are used for controlling the video output circuitry. There are 8 pins dedicated to the cursor for the 8 plane frame buffer.

The BTEN L and LDMSEL H pins are required for writing the RAMDAC. RAMDAC Look-Up Table and control registers are written from the Video Shift Register. Both RAMDAC reads and the video out circuitry MUST be read through a Diagnostic register.

LCG Signal Name	TYPE	Description
BLANK L	BT4	Video Blank Signal for 8 plane
SYNC L	BT4	Video Synch Signal
VSHIFT L	BT4	Video Shift Register Enable
MSEL0 H	BT4	Video Multiplexer Select
BTEN L	BT4	BT458 Enable Signal for LUT Loads
LDMSEL H	BT4	BT458 LUT Input Multiplexer Select
NIBCLK H	DRVT4	Video Nibble Clock - In
CURSOR<7:3> H	BT4	Video Cursor/1 Plane Video I/O
CURSOR<2:0> H	BD4C	also used for the MV_Dal Accelerator Interface

TOTAL Signals = 15 pins

10.1.5 MV_DAL Accelerator Interface Pins - 8+ Pins

The Cursor<7:0> H pins are reassigned to interface with 3D Scan Proc and the Imaging Accelerator. An accelerator can be designed with an 8 plane LCG Video Module, but the LCG Video Module will need to upgrade the RAMDAC to include the Cursor, such as the Bt459.

10.1.5.1 Accelerator Interface Functionality Levels:

- Program I/O: CPU Accesses: longword reads and up to quadword writes
- LCG Pseudo DMA: LCG transfers data between Accelerator and Memory

10.1.5.1.1 Program I/O

The MV_DAL Option is exactly like a LCG Frame Buffer with additional support for Stalling on reads and writes, as well as, an Interrupt Line which is included in the LCG Interrupt/Error Register with its enable in the LCG Interrupt Enable Register.

10.1.5.1.2 LCG Pseudo DMA

This allows the CPU to set up one or more data transfers between memory and the Accelerator through the LCG Command Packets. The data movement can be in the DECwindows Server's Virtual Process Space.

The ROP_RECT Command Packet can be used to allows the Address Generator to continually execute Memory Reads with a data size Longword or Octaword. Two pins defined below signal the Accelerator the data size present on the MV_DAT<31:00> signals. This allows the data to be read from memory and transfered to the accelerator without moving it throught the S-Chip. This effectively doubles the number of commands fed to the Accelerator with the same percentage of memory bandwidth used. The LEN_AB<1:0> H signals latch the data into the option when the data is valid on the MV_DAT.

10.1.5.2 MV_DAL Option Pins (Cursor<7:0>):

```
Program I/O Required Pins:

 CLKP<3> H.........(OUT) - CURSOR<7>
 OPTION RESET H ...(OUT) - CURSOR<6>
 CLKP<1> H.........(OUT) - CURSOR<5>
 MV_DAL INTERRUPT L (IN) - CURSOR<1>
 MV_DAL STALL H ....(IN) - CURSOR<0>

LCG Pseudo DMA Additional Required Pins:

 DATA 0 VALID H ...(OUT) - CURSOR<4>
 DATA 1 VALID H ...(OUT) - CURSOR<3>
 MV_DAL INHIBIT H ..(IN) - CURSOR<2>
```

10.1.5.3 MV_DAL Accesses

The MV_DAL is a RAS/CAS Address bus and follows all of the normal rules of memory accesses with RAS/CAS addressing. See description of the Memory System and pins required for Memory and Frame Buffer Accesses.

10.1.5.3.1 MV_DAL Reads

Main Memory and Frame Buffer Reads return data at the beginning of the second cycle after RAS is asserted. Data is latched into the S-Chip during CLKP4 H. The LEN_BA L signal is used to open the 74F543 bidirectional latches which are assumed to be on all MV_DAL interfaces. The timing constraints of a 74F543 must be met.

Cycle 0 (60-80ns)	Cycle 1	Cycle 2	Cycle 3	
RAS Address Plane Mask	CAS Address Assert Stall (Prog TimeOut)	Read Data 0 Read Data 0 Data 0 Valid	Read Data 1 Read Data 1 Data 1 Valid	... ...

10.1.5.3.2 MV_DAL Writes

Main Memory and Frame Buffer Writes accept data with the LEN_AB<1:0> L signals. The Write Mask is present during the cycle that RAS is asserted. Starting on the next cycle, data is present on the MV_DAL for up to 4 cycles with octaword writes. Data is clocked internal to the S-Chip just before the MV_DAT transceivers on the rising edge of P2.

Cycle 0 (60-80ns)	Cycle 1	Cycle 2	Cycle 3	
RAS Address Plane Mask	CAS address Write Data 0 Assert Stall (Prog TimeOut)	Write Data 1	Write Data 2	... ...

10.1.5.4 S-Chip Clock H (CLKP3 H) Signal

This signal is a buffered CLKP<3> H. All interface signals MUST use the rising edge of this signal for registered communications. This will give enough time to decode the signal and set it up for registering it on the next rising edge of CLKP<3> H internal to the chip.

10.1.5.5 S-Chip Clock H (CLKP1 H) Signal

This signal is sent for higher granularity on the System clocks.

10.1.5.6 Option RESET L Signal

This signal will be asserted on system reset, as well as, when the Option Reset bit is written with a 1 in the LCG Graphics Control Register <21> Writing Bit <21> results in a 1 to 2 video scan line length Reset signal being sent to the MV_Dal Option.

10.1.5.7 MV_DAL Interrupt Signal

This signal can be polled in bit <18> the LCG Graphics Interrupt/Error Register, GRAPHICS_INT/ERR. This signal is also passed on to the S-Chip interrupt logic and it has its own interrupt vector.

10.1.5.8 MV_DAL Stall Signal

This signal will be used for both reads and writes. If the MEM_STALL_MASK bit is asserted (1 bit per frame buffer memory bank, 4 in total), then the memory controller samples stall when waiting for read data or when presenting data on writes. Reads to a slow memory bank are always stalled at least 1 cycle. Writes are always stalled 1 cycle if the MEM_STALL_ON_WRITE bit is a 1. The memory access continues the next cycle after the Stall signal is deasserted.

The S-Chip stalls for a maximum of between 1 and 5 scan lines (4 programmed options) of time before aborting the access, resetting the option, and asserting the Option Interrupt to the CPU.

The MV_DAL accelerator MUST NEVER be in the FIFO FULL State where it asserts the Stall signal when a write is directed to the FIFO with a Program I/O interface. Using the Stall signal is strongly discouraged, since it stalls the whole system.

10.1.5.9 MV_DAL Inhibit Signal

This signal is used for signaling the LCG Address Generator Arbitration logic that the MV_DAL Option cannot accept Data on Writes nor Supply Data on Reads, depending on the Context. This signal will mask out the LCG Address Generator's arbitration line to the Memory System. When this signal is deasserted, the Address Generator will be allowed to arbitrate for memory.

10.1.5.10 DATA 0/1 VALID Signal

The purpose of these signals is to reduce the MV_DAL accesses to a read from memory to the Option versus reading to the LCG Graphics Data Buffer and then Writing back to the MV_DAL Option. This effectively doubles the available bandwidth. Writes from the MV_DAL Option must be buffered within the Graphics Data Buffer.

These signals are used for signaling the size of the data present on the MV_DAT<31:00> signals from memory which is intended for the Option. The LEN_AB<1:0> L signals latch the data into the Option. CLKP3 H should be used as the timing signal to qualify LEN_AB<1:0> L for latching the data.

MV_VALID <1:0> H		Operand Length
0	0	Idle
0	1	Longword
1	0	Quadword
1	1	Octaword

10.2 Internal Interface Signals

The following signals are used to communicate from LCG to the rest of the S-Chip, as well as, the major buses within the LCG subsystem.

10.2.1 PVAX2 H / MARIAH L - PV2 H (IN)

This signal indicates that the memory system must change the parity used on memory and the address range.

	Parity	Address Range
PVAX2	Byte Dependant	32M byte Max
PMARIAH	All ODD	104M byte Max

When PV2 H is asserted then odd parity is on odd byte addresses and even parity is on even byte addresses. Also, all memory addresses above 32M bytes are ignored and the RAS signals are masked so that no memory access occurs.

10.2.2 External Controller Request - NI_/SC_REQ<2:0> H (IN)

The Request Valid bit is asserted by the external controller whenever a valid address, command, and access size is present for the LCG subsystem. Memory accesses support up to quadword.

Bit Position	Signal Name
2	Request Valid H
1	Read H / Write L
0	Access Size

Binary Code	Access Size
0	Longword
1	Quadword

10.2.3 CPU (SOC) Request - SOCREQ<3:0> H (IN)

Bits <2:0> are exactly the same as above. Bit 3 indicates that the write is to the COMMAND FIFO. This is needed for timing considerations, since the Command FIFO Controller must supply the actual memory address.

COMMAND FIFO Writes are a special case for addressing main memory. When a Command FIFO Write is requested from the CPU and Short Circuit is NOT present, the address<3:2> within an Octaword is selected from the CPU and the rest of the address<26:04> is selected from the Command FIFO Base and Tail. All Byte Masks and command bits are selected from the CPU. If the Comand FIFO and Virtual Translation state machine is not currently involved in a memory access, the Command FIFO and Virtual Translation subsystem is forced to select the Command FIFO write address when ever SOCREQ3 is asserted high.

When in Short Circuit mode, the Command FIFO write is written to the Command FIFO Residue latches. The data is also written to the appropriate registers/latches.

```
Short Circuit = (FIFO Empty)(Clip List OFF)(Address Generator NOT BUSY)
```

10.2.4 DMA Address Filter Stall - FSTALL L (IN) - SOC Int.

This signal is asserted by the DMA Address Filter when all of its address buffers are full and another address cannot be accepted. When FSTALL L is asserted, it masks (disables requests into the arbitration priority encoder) writes from:

- NI Writes
- SCSI Writes
- Address Generator Writes - not including Frame Buffer Writes
- CPU Reads

10.2.5 External Address In - SY,NI,SC_ADR<26:02> H (IN)

Contains the address of the memory or device to be accessed by the NI, SCSI, or SOC. Physical address range of <26:02> is a 128M bytes of memory space maximum (3FF,FFFF = 134,217,727). Data and Byte Mask are selected by the External Select signals, EXTSEL<1:0> H. The address busses are multiplexed within the memory controller to provide the minimum propogation time to the memory system.

10.2.6 I/O Address Access - IO_SOC H(IN)

If IO_SOC H is asserted HIGH, then the CPU's access is an I/O access. If LOW, then the access is to memory. Neither the NI nor SCSI can access I/O space.

10.2.7 External Data In - DATAIN<31:00> H (IN)

Contains the data from one of the external controllers. The WDSEL H signal selects the appropriate longword on writes. The current interface supports up to quadword writes.

10.2.8 External Controller Byte Mask - EXBMSK<7:0> H (IN)

These signals are also sent by one of the external controllers, dependent on the EXTSEL<1:0> H signals. The byte mask is for up to naturally aligned quadword writes. Each bit write enables a byte. Bit 0 enables the writing of address 0. The entire quadword byte mask is sent from the external controller's write buffer.

10.2.9 System RAM Read - SY_RAMRD L (IN)

This signal is asserted LOW whenever a read to main memory space occurs by the CPU. The memory controller uses this signal to bypass the P1 gating on the transceiver output enables to main memory.

10.2.10 External Controller Select - EXTSEL<1:0> H (OUT) - Flow S.M.

These signals are the multiplexer selection bits for the External Controller's Address, Data, Mask Bits, and Request. They are also be used to identify to the Flow Control state machine the intended controller to be acknowledged. The table below indicate the expected selects for the multiplexer. These signals are buffered REQ_SEL<1:0> H which originate from the Memory Controller State Machine's Next Request latch. This latch is open before an access is complete for allowing the next access to start.

Binary Code	Longword Number
0 0	Internal - AG or Video
0 1	NI Controller
1 0	SOC Interface
1 1	SCSI Controller

10.2.11 Data Ready - SY,NI,SC_DRDY<1:0> L (OUT) - Flow S.M.

During READ accesses, DRDY0 L is used for indicating that longword address 0 data returned is valid. DREADY1 L indicates that longword address 4 data is valid. The order of the data returned is dependent on ADRS<2> H.

There are 3 sets of DRDY<1:0> L signals for the 3 external memory masters. EXTSEL<1:0> H cannot be used, since the address for the next memory access is pipelined with returning the read data. The signal names are listed below.

External Controllers DREADY<1:0> L
NI_DRDY<1:0> L
SC_DRDY<1:0> L
SY_DRDY<1:0> L

These signals are asserted during the cycle(P1) that data is valid on DALOUT<31:00> H. If there is a parity error, MEMERR L is asserted when the data is returned and should be registered on the rising edge of DRDY L.

10.2.12 Write Data Select - WDSEL H (OUT) - Mem S.M.

When WDSEL H is 0, longword 0 is selected. When WDSEL H is a 1, then longword 1 is selected for writing data. This signal should control the write buffer 2 to 1 multiplexer directly. EXTSEL<1:0> H selects the appropriate external controller.

10.2.13 Mono - MONO L (OUT) - Memctl

MONO L is derived from the monochrome bit in the MEMORY_CONFIG register.

10.2.14 External Address Acknowledge - EXAD_ACK H (OUT) - Mem S.M.

This signal is asserted during State 1 of the Memory Controller State Machine. This means that the exteral address has been latched and that the access has started. This is created from the Memory Request Acknowledge signal, MREQACK H, which originates from the Memory State Machine. EXTSEL<1:0> H selects the appropriate external controller.

The access request must be deasserted the cycle after receiving this signal. This is to insure that the next request prioritization can occur in parallel with the present access.

10.2.15 External Access Done - EXACDON<2:0> H (OUT) - Flow S.M.

This signal is asserted when all of the access data has been transfered. It is asserted HIGH when all of the data is buffered within the Pixel SLU's Write Data Path for Writes or after all of the data has been returned on the DALOUT<31:00> H for Reads. This is created from the Memory Access Done signal, MACDON H, which originates from the Flow State Machine. This signal is created by watching the Memory Controller Signals which signal

that read (Memory Data State) and write data (Memory Data Control) are present. These 3 signals are created from the the EXTSEL<1:0> signals and the MACDON H signal. The EXTSEL<1:0> signals cannot be used directly, since they may change before the last read data is returned. The next access starts before all data is returned to the requesting master.

Bit Position	Signal Name
2	SCSI CONTROLLER ACK H
1	SOC INTERFACE ACK H
0	NI CONTROLLER ACK H

10.2.16 Address Valid - AD_VAL L (OUT) - Flow S.M.

This signal is asserted LOW whenever a write or read to main memory or I/O space occurs by any external controller, Address Generator, Virtual Translation subsystem, Command FIFO, Clip List, or Video subsystem. A valid address is on the DALOUT<26:02> H signals when this signal is asserted.

This signal is asserted LOW on P1 and is used as the latch enable.

10.2.17 Access Read / Write - AC_READ H (OUT) - Flow S.M.

When this signal is asserted high the access is a read. When low, the access is a write.

10.2.18 Access Size - AC_SIZE<1:0> H (OUT) - Flow S.M.

These signals indicate the size of the access in progress.

Binary Code	Operand Size
0 0	Longword
0 1	Quadword
1 0	Octaword
1 1	Word - Monochrome Video

10.2.19 Address/Data Out - DALOUT<31:00> H (OUT) - AD_OUT

This is the Address/Data bus which sends data to the External Controllers, DMA Filter, and PTE CAM. Address and Data are valid during the middle of P1 to P4 when ever either AD_VAL L or DREADY<1:0> L are asserted.

10.2.20 Memory Read Back - MAD_RB<31:00> H (OUT) - AD_OUT

Data from the LCG read back paths are muxed on this bus.

10.2.21 Parity Out - PARITY<3:0> H (OUT) - MDAL_INT

These are the parity bits which are stored in main memory which are passed back to the external controllers on reads. The parity bits are valid at the same time that Data is valid on the DALOUT<31:00> H bus.

10.2.22 Parity Error - MEM_ERR L (OUT) - FLOW S.M.

This bit indicates that there is a parity error on the data returned on DALOUT<31:00> H. MEMERR L is combinational and must be registered on the rising edge of DREADY L. The memory system latches the last address which caused the parity error, but it is up to the controller which requested the data to report the error through either an interrupt or directly in the case of CPU reads.

10.2.23 I/O Access - IO_ACC H (OUT) - AD_OUT

This signal is indicates that the address on the DALOUT<31:00> H signals is either an I/O address (asserted high) or a memory address (low). It is valid when AD_VAL L is asserted.

10.2.24 Graphics/Video Interrupt - VAGIRQ H (OUT) - Virt S.M.

This signal is asserted when ever any exception or error occurs within the Graphics, Virtual Translation Exception, Memory, or Video system. This signal remains asserted high until all interrupts in the Graphics Interrupt and Error Status have been cleared. It reasserts with the next Interrupt condition which occurs (rising edge).

10.2.25 Test Control - SY_TEST L (IN)

Assertion of SY_TEST L will configure the chip to be in the test mode. This pin must be deasserted for normal operations.

10.2.26 Test Mask - SY_TESTI<2:0> H (IN)

These signals in conjunction with the SY_TEST L pin configure the test mode.

Binary Code	Operand Size
0 0 X	Open
0 1 X	LCG Test
1 0 X	System Test
1 1 X	Open

10.3 LCG Internal Control Signals

This group of signal definitions are the control signals which communicate between the major logical functions of the LCG subsystem.

10.3.1 Address Generator Write Byte Mask - AGBMSK<7:0> H - Write M.G.

Theses signals contain the byte masks for up to a quadword. The Write Mask Generator creates the byte masks for up to an octaword. When MDATCTL8 H = 0, the least significant quadword's write mask is asserted. When MDATCTL8 H =1, the most significant quadword's write mask is asserted. When writing only a longword or less, AGBMSK<3:0> H contain the corresponding byte masks.

10.3.2 Address Generator Memory Request - AGMREQ<7:0> H - AG S.M.

Bit Position	Signal Name
7	Virtual H / Physical L
6	Busy H
5	Arbitrate H
4	Read Operand Ident: Source H / Stencil L
3:2	Access Type
1:0	Operand Size

10.3.2.1 Access Type Field

Binary Code	Signal Name	
0 0	Write	- Destination
0 1	Read	- Source or Stencil
1 0	Read-Modify-Write	- Destination
1 1	NOP	

10.3.2.2 Operand Size Field

Binary Code	Operand Size
0 0	Longword
0 1	Quadword
1 0	Octaword
1 1	NOP

10.3.3 Memory Address Request - MA_REQ<9:0> H - Virt S.M.

This bus comes from the Virtual Translation/Command FIFO/Clip List state machine. This state machine also intercepts or passes on the Address Generator's memory request, depending on whether the address is physical or virtual. If a virtual request results in a PTE Cache Miss, then the state machine either replaces the PTE entry with the correct entry or signals a problem with doing this through the VIRTIRG H signal. The state machine can only generate read requests EXCEPT when writing the Command FIFO residue latches to the Command FIFO when the Clip List is enabled.

The 1st Command Packet Longword signal, MA_REQ8 H, forces the Flow Control State Machine that the next longword from either a Command FIFO or Clip List access is the first longword in a packet. The Flow Control forces FCTL<16> to 1 and latch the Command Opcode from PXDAT<31:24> H. All registers which can be written from the Command FIFO on the first longword of a packet MUST decode the opcode from PXDAT<31:24> H during the cycle that FCTL<16> is high.

The Command FIFO State Machine latches the number of longwords in the command packet from PXDAT<23:22> H, as well as the Command Opcode. When all longwords within a command packet have been read and the command packet is not an Action Command Packet (Address Generator, Command FIFO, or Clip List becomes active), the Command FIFO asserts the 1st Command Packet signal again. The Command FIFO continues to issue Octaword Reads until it encounters an Action Command Packet. The Command FIFO decodes the Action Command Packet Opcodes.

When an Action Command Packet is encountered, the whole Command Packet is read and all remaining longwords within an octaword is stored within the Command FIFO Residue latches. The Command FIFO signals the Flow Control that the next longword read is an Invalid Command Packet Data with bit 9. The Flow Control signals all registers/latches to ignor the data on the PIXEL DATA bus by not asserting any of the Byte Mask signals (0).

The data following an Action Command Packet within an octaword is latched into the Command FIFO residue buffer for Command FIFO requests and discarded for Clip List requests (Clip List is padded for Octaword aligned accesses).

Bit Position	Signal Name
9	1 = Valid Command Packet Data
8	1 = 1st Command Packet Longword
7	Arbitrate H
6:4	Operand Identification
3:2	Access Type (Same as AGMREQ)
1:0	Operand Size

10.3.3.1 Operand Identification Field

Binary Code	Source/Destination
0 0 0	Address Generator - Source 2 (Stencil)
0 0 1	Address Generator - Source 1
0 1 0	Address Generator - Destination
0 1 1	Virtual Translation Subsystem
1 0 0	Command FIFO - Memory Octaword Read
1 0 1	Clip List - Memory Octaword Read
1 1 0	Command FIFO - Command FIFO Residue Read
1 1 1	Command FIFO - Command FIFO Residue Read

10.3.3.2 Access Type Field

Binary Code	Signal Name	
0 0	Write	- Destination
0 1	Read	- Source or Stencil
1 0	Read-Modify-Write	- Destination
1 1	NOP	

10.3.3.3 Operand Size Field

The Command FIFO requests an octaword when 3 valid Command FIFO residue latches are valid. The last longword of the octaword request is signaled to be Invalid Data using MA_REQ9. The Command FIFO selects the appropriate residue latch, while the Flow Control selects the Command FIFO residue latches for the PIXEL SLU input multiplexer. Quadword and longword requests are made when there is 2 and 1 longword valid within the Command FIFO Residue latches.

The Command FIFO must also assert the appropriate size and byte masks for writing the Command FIFO Residue latches to memory.

Binary Code	Operand Size
0 0	Longword
0 1	Quadword
1 0	Octaword
1 1	NOP

10.3.4 Video / Memory Refresh Memory Request - V_M_REQ<2:0> H - Vid S.M.

Memory Refresh asserts both main and video memory RAS signals. All of main and video memory is RAS-only refreshed simultaneously.

The Octaword Read for the Cursor Loads is always to main memory.

The Video Shift Register Loads asserts all VRAM RAS signals.

Bit Position	Video Request Code
2	Video Request H
<1:0>	Video Request Type:
0 0	Memory Refresh
0 1	Read Octaword (Cursor Load)
1 0	Video Shift Register 1/2 Load
1 1	Video Shift Register Load

10.3.5 Next Memory Request - NXTMREQ<5:0> H - Flow S.M.

Bit Position	Signal Name
5:4	Requester Identification
3:2	Access Type
1:0	Operand Size

10.3.5.1 Requester Identification Field

Binary Code	Requester Identification
0 0	Video Subsystem
0 1	External Commander
1 0	Address Generator
1 1	No Request Present - Register Accesses also

10.3.5.2 Video Access Type Field Definition

Binary Code	Video Access Type
0 0	Memory Refresh
0 1	Read Octaword (Cursor Load)
1 0	Video Shift Register 1/2 Load
1 1	Video Shift Register Load

10.3.5.3 External/Address Generator Access Type Field Definition

Binary Code	Ext/AG Access Type
0 0	Write
0 1	Read
1 0	Read-Modify-Write
1 1	NOP

10.3.5.4 Operand Size Field

Binary Code	Operand Size
0 0	Longword
0 1	Quadword
1 0	Octaword
1 1	NOP

10.3.6 Memory Type - MEMTYP<2:0> H - MEMCTL

Binary Code	MV_DAL Memory/Device Type
0 0 0	Main Memory
0 0 1	8 Plane Frame Buffer
0 1 0	No Memory Access
0 1 1	Monochrome Frame Buffer Longword
1 0 0	System / Video ROM
1 0 1	Video Subsystem
1 1 0	No Memory Access
1 1 1	No Memory Access

10.3.7 Prioritorized Request Identificaiton - PREQSEL<2:0> H - Flow S.M.

Binary Code	Requesting Master
0 0 0	Video Subsystem
0 0 1	NI Controller
0 1 0	SOC Interface
0 1 1	SCSI Controller
1 0 0	AG/Virtual Controller
1 0 1	NO REQUEST PRESENT
1 1 0	AG/Virtual Controller - Command FIFO Write
1 1 1	NO REQUEST PRESENT

10.3.8 Request Identification - REQ_SEL<2:0> H - Mem S.M.

This field is a latched version of the Prioritorized Request Identification field above. The latch is open during the last state of the memory state machines access flow and remains open during state 0 if no accesses are present.

10.3.9 OTF Access - OTF H (OUT) - MEMCTL

This signal indicates that the access is an on the fly read. OTF has the same timing as the MRAS signal.

10.3.10 Write Count - WLWCNT<2:0> H (OUT) - MEMCTL

These bits indicate valid write data on the PXDAT<31:00> H. WLWCNT<2:0> H changes on P3.

Binary Code	Operand Size
0 X X	No Valid Data
1 0 0	Data Valid - LW0
1 0 1	Data Valid - LW1
1 1 0	Data Valid - LW2
1 1 1	Data Valid - LW3

10.3.11 Memory Request Acknowledge - MREQACK H - Mem S.M.

This signal indicates that the requester's address is no longer needed. The access is in progress.

10.3.12 Memory Busy - MBUSY H - Mem S.M.

This signal indicates that the memory state machine is busy. MBUSY H is asserted for the entire access.

10.3.13 Memory Access Done - MACDON H - Mem S.M.

This signal indicates that the access is done. MACDON H is asserted in the last state of the access.

10.3.14 Memory Data State - MDSTAT<2:0> H - Mem S.M.

These signals are asserted 2 cycles before the incoming longword from Memory. They are pipelined 1 cycle by the Flow Control State Machine and asserted as part of the FCTL<16:00> H bus.

Bit Position	Signal Name
2	Data Valid
1:0	Longword Number

10.3.15 Memory Data Control - MDATCTL<8:0> H - Mem S.M.

All signals below are generated by the Memory State Machine every long cycle, CLKP3 H (60 to 80 nsec). Bits<8:4> are sent to the Pixel Slu, while bits <3:0> are sent to the MV_DAL interface logic sections.

The Destination Latch Enable signals gate the CLKP1 H clock. Bit 6 enables the least significant longword latch, while bit 7 enables the most signficant. Bit 7 is also buffered to create WDSEL for the external controllers write buffer data select.

The MV_DAL Output Enable signal is clocked in the MVDAL_INT logic section every short cycle, CLK24 H (30 to 40 nsec). This allows the MV_DAL to be active for as short as 30ns for avoiding tri-state overlap between multiple drivers. The output enables are turned on with the CLKP2 H edge and turned off with the CLKP4 H edge.

The MV_DAL OUT Clock is active at the end of each long cycle, CLKP2 H, if enabled.

The MV_DAL IN Clock is active at the middle of the long cycle, CLKP4 H.

Bit Position	Signal Name
[illegible]	0 = Quadword 0, 1 = Quadword 1
7:6	DST<1:0> Latch Enable [illegible]
5:4	DST<1:0> Multiplexer Selects H
3:2	MV_DAL Output Enable<1:0> H
1	MV_DAL IN Clock Enable H
0	MV-DAL OUT Clock Enable H

The Destination Latch Enable <1:0> signals:

Binary Code	Longword Number
0 0	No Latch Enable
0 1	Destination Latch 0
1 0	Destination Latch 1
1 1	No Latch Enable

The Destination Multiplexer Enable <1:0> signals:

Binary Code	Longword Number
0 0	Pass Destination Latch 0
0 1	Pass Destination Latch 1
1 x	Pass the Plane Mask

The outputted Plane Mask is dependant on the Force Write Mask Plane Mask signal, AST_PMSK H, described below.

10.3.16 Memory Interface Signals - M_SIGS<21:00> H - Mem S.M.

All signals below are generated by the Memory State Machine every long cycle, CLKP3 H(60 to 80 nsec). The actual signals which the MV_DAL and Memory system receive are clocked every short cycle, CLK24 H(30 to 40 nsec). Bit 0 is asserted during the first short cycle, CLKP4 H, and bit 1 is asserted during the second short cycle, CLKP2 H, of a long cycle.

MBAOE<1:0> H and VBAOE<1:0> H (for 8 planes) or VABOE H (for monochrome) are the MV_DAL output enable control signals. They are NANDed with CLKP3 L before going off of the chip. This turns off all output drivers for the first quarter cycle for avoiding tristate overlap. MABOE H and VABOE H (for 8 planes) or VBAOE<1:0> H are enabled for full cycles, since they drive the DRAM and VRAMs directly and tristate overlap is not a problem here.

Bit Position	Signal Name
21:20	RAS <1:0> H
19:18	CAS <1:0> L
17:16	BWRT <1:0> L
15:14	DT <1:0> L
13:12	DSF <1:0> L
11	MBAOE1 H
10	MBAOE0 H
09	MABOE H
08	VBAOE1 H
07	VBAOE0 H
06	VABOE H
05:04	LEAB1 <1:0> L
03:02	LEAB0 <1:0> L
01:00	LEBA <1:0> L

10.3.17 Memory Control - MEM_CTL<4:0> H - Mem S.M.

Bit Position	Signal Name
4:3	Byte Mask Latch Enables H
0 0	NOP
0 1	Load Longword Enables Only
1 0	Load Quadword Enables
1 1	Read - Force all Byte Enables Asserted
2	Memory Address Multiplexer Select: RAS H/CAS L
1	Toggle CAS Address H
0	Latch Physical Address H

10.3.18 Pixel Slu Select - P_SLU_I<1:0> H (OUT) - MEMCTL

This bus selects from different input sources in the PIXELSLU. P_SLU_I<1:0> H changes on P4.

Binary Code	Operand Size
0 0	Select Data Buffer
0 1	Select DATAIN
1 0	Select MEMDAT
1 1	Select Residue Buffer

10.3.19 Internal Access Done - IACDON<1:0> H (OUT) - Flow S.M.

This signal is asserted when the address and request lines are not required any more. It is asserted HIGH the cycle after the address and request has been latched. IACDON is a registered signal which is clocked on the rising edge of CLKP<3> H. The memory master must lower its request either during the cycle that IACDON is asserted or the cycle after.

Bit Position	Signal Name
1	AG, CF, & Virt ACK H
0	VIDEO/MEMORY REFRESH SYSTEM ACK H

10.3.20 AG Longword Enable - AGLWEN<1:0> L - MEMCTL

AGLWEN<1:0> L pass address bit 2 to the AG. This bus is asserted during the access.

10.3.21 Frame Buffer Select - FB_SEL<1:0> H (OUT) - MEMCTL

The monochrome bit and the frame buffer size bit are buffered to generate this bus.

Binary Code	Operand Size
0 0	Monochrome, 256kX4 frame buffer
0 1	Monochrome, 128kX8 frame buffer
1 0	Color, 1mX1 or 256kX4 frame buffer
1 1	Color, 4mX1 or 128kX8 frame buffer

10.3.22 Flow Control Bus - FCTL<16:00> H - Flow S.M.

The definition of this control bus is dependent on bit 15. Command Packet and Clip List Reads from memory use the Internal Register Access format of the FCTL<16:00> H Bus.

10.3.22.1 Flow Control Bus - Internal Register Accesses

When bit 15 is asserted high, then an interal register is either being written or read and bits <14:00> are interpretted as follows:

Bit Position	Signal Name
16	Command FIFO/Clip List Access H
15	IGR Load H
14	Read H / Write L
13:10	Byte Mask H
09:00	Register Address

This signal definition is used for accessing interal registers from either the Command FIFO or from an external source, such as the CPU. The register address is decoded by each logic subsection from bits <09:00>. When any one of the byte mask bits are asserted high, then data is valid. The byte mask bits are used for loading multiple internal registers and latches individually or together.

10.3.22.1.1 Register Address Field

This field is made up of the following fields when the register is loaded from either the Command FIFO or the Window Buffer. During the first longword of a Command Packet the Command Opcode field is forced to all zeros.

Bit Position	Register Address
09:02	Command Opcode = Address<11:04> H
01:00	Longword Number = Address<03:02> H

10.3.22.1.2 Longword Number Field

This field is used to indicate the longword number within a Command Packet when reading from either the Commad FIFO or Clip List. The Control for clearing this field comes from the Command FIFO state machine.

Binary Code	Longword Number
0 0	1st Longword
0 1	2nd Longword
1 0	3rd Longword
1 1	4th Longword

10.3.22.2 Flow control Bus - MV_DAL Accesses

The Flow Control Signals are defined as follows when bit 15 is asserted low: These signals are valid for Reads from the MV_DAL to the opeand field. Writes occur only from the External Controllers and the Address Generator. The MDATCTL control bus is used for controlling writes. This bus communicates to the PIXEL SLU, Address Generator, and the Flow Control for communication with the External Controllers.

Bit Position	Signal Name
16	Undefined = 0
15	Memory Access L
14	Read H / Write L
13	Read Data Valid H
12:10	Operand Source/Destination
09	0 = IDLE 1 = Memory Access
08:04	Undefined = 0
03:02	Operand Size
01:00	Longword Number

10.3.22.2.1 Operand Source/Destination Field

Binary Code	Source/Destination Operand
0 0 0	Address Generator - Stencil
0 0 1	Address Generator - Source
0 1 0	Address Generator - Destination
0 1 1	Virtual Translation Subsystem
1 0 0	NOP
1 0 1	NOP
1 1 0	Video Cursor
1 1 1	External Controller

10.3.22.2.2 Operand Size Field

Binary Code	Operand Size
0 0	Longword
0 1	Quadword
1 0	Octaword
1 1	NOP

10.3.22.2.3 Idle Bit

Bit Position	Signal Name
9 = 0	Idle
9 = 1	Memory Access Active

10.3.23 Mask Generation Control - MSKCTL<2:0> H - AG S.M.

Bit Position	Signal Name
2	Indicates that a "Start" write mask should be created. This bit should be asserted on the first write of every scanline.
1	Indicates that an "End[Clip]" mask should be created. This signal should be true whenever the X1Clip indicates an "almost out" condition.
0	Indicates that an "End[Width]" mask should be created. This bit should be asserted whenever the width decrement indicates an "almost out" condition.

10.3.24 Shift Control Bus - SHF_CTL<11:00> H - AG S.M.

Bit Position	Signal Name
11	Indicates that valid data is on the bus
10:9	Indicate the type of data that is on the bus: 00 - S1 Shift Count 01 - S2 Shift Count 10 - Mask Generator Start Point 11 - Mask Generator End Point
▮	Indicates whether the destination is single plane or eight plane 0 - Dest is Eight Plane 1 - Dest is Single Plane
If shift data is on the bus,	
7:0	The shift count to be taken by the SLU or by the Graphics Data Buffer.
If Write Mask data is on the bus,	
7	Indicates the direction of the operation 0 - Left to Right 1 - Right to Left
6	Indicates whether the End Point being transmitted is for Clipping or for Width. 0 - Endpoint is from width 1 - Endpoint is from clipping
5:0	Start or End Point to use in Mask generation

10.3.25 Graphics Data Buffer Control Bus - DBCTL<7:0> H - AG S.M.

Bit Position	Signal Name
7:6	Indicate the packing of the source: 00 - Source is MultiPlane 01 - Source is Single Plane 10 - Source is Packed Text 11 - Undefined
5	Indicates the packing of the data leaving the GDB: 0 - Data is MultiPlane 1 - Data is Single Plane
4:3	Indicates the buffer quarter to be used for data in.
2	Indicates the buffer half to be used for data out.
1:0	The two lower bits of the last destination address generated. These bits are bits <3:2> of the BYTE address, or bits <1:0> of the LONGWORD address.

10.3.26 Pixel SLU Control Bus - SLUCTL<3:0> H - AG S.M.

TBS - Blaise

10.3.27 Command FIFO Residue Control Signals - CFRCTL<5:0> h - Virt S.M.

These signals originate from the FIFO controller which is part of the Memory Address Multiplexer logic section. They control the Command FIFO Residue Buffer, 3 longwords which resides within the Pixel SLU.

Bit Position	Signal Name
5	Longword 3 Mux Enable H
4	Longword 2 Mux Enable H
3	Longword 1 Mux Enable H
2	Longword 3 Latch Enable H
1	Longword 2 Latch Enable H
0	Longword 1 Latch Enable H

10.3.28 Short Circuit - SCIRCUIT H - Virt S.M.

This signal indicates that a Command FIFO write from the CPU does not go to memory, but directly to the LCG registers and the Command FIFO Residue Buffer. The FCTL bus signals a register write and supplies the Command Opcode and Longword Number.

```
SCIRCUIT = (FIFO EMPTY)(CLIP LIST DISABLED)(NOT(ADDRESS GENERATOR BUSY))
```

10.3.29 Address Generator Acknowledge - AG_ACK H - Virt S.M.

The Virtual State Machine sends the access acknowledge to the Address Generator. It is asserted after Address Acknowledge is received for reads generated from the Address Generator. For writes it is generated after the Virtual State Machine receives Access Done.

10.3.30 Pixel Function Requires Read-Modify-Write - PFCNRMW H - Pixel SIU

This signal is a combination decode of the current Boolean Pixel Function that require a boolean operation to be performed between the source and destination graphics operands. This signal is asserted whenever transparency or a Stencil operand is active.

10.3.31 Plane Mask Equals All Planes Asserted - PMEQSET H - Write M.G.

This signal is asserted whenever all of the bits in the Plane Mask Latch are asserted. This is used when the address generator's destination operand is in main memory and enables a write instead of a read-modify-write. operation to main memory.

10.3.32 Assert Plane Mask - AST_PMSK H - Mem Adrs Mux

This signal is asserted high whenever an access is made to the frame buffer address range which requires the write mask. Whenever deasserted, all 1s are asserted. This signal originates from the Memory Address Multiplexer and is sent to the Pixel SLU.

11 LCG Registers

This section describes the LCG registers.

11.1 Register Addresses

The LCG register address space is architecturally restricted to physical addresses in the range of 2010.0000 to 2013.FFFF. The registers are arranged in this space according to the following table:

Address Bits	Description
31:16	These bits are always in the range 2010 to 2013
15:12	If any of these bits are set, this is a restricted register, which means that it may not be written from the command FIFO.
11:04	These bits are the command opcode of the command packet which is normally used to write this register from the FIFO. Registers which are not associated with a command packet use an unused opcode for these bits. These bits are equivelent to bits 09:02 on the flow control bus.
03:02	These bits are equal to the longword number of the register's value in it's associated command packet.
01:00	These bits specify the byte address of the register.

All the LCG registers are longword access. The CPU has read/write access to all of the registers. Registers may be protected on a page-by-page basis for a particular process by using the standard PTE-based memory access protection supported by VAX/VMS. The register addresses have been arranged to facilitate this form of CPU access protection.

The command FIFO may read any register but can write only some of them. Registers which have any one of the bits <15:12> in their address set may not be written by command packets in the FIFO.

11.2 Register Indexes

The register index for a given register may be obtained by extracting bits <17:02> from the registers address. The register index is used by the LOAD_REGISTER and STORE_REGISTER commands.

11.3 Internal Address Decode Control

The following table comes from the "LCG Internal Control Signals" section. It describes the signals on the Flow Control Bus during internal register accesses.

Bit Position	Flow Control Bus Definition
16	1 = Command FIFO or Clip List Access
15	1 = Internal Register Access
14	1 = Read/0 = Write
13:10	Byte Mask (Any Bit Set (1) = Data Valid)
09:02	Command Opcode/Register Address<11:04>
01:00	Command Packet Longword Number

11.4 LCG Register List Description

The following table is a complete list all of the LCG registers.

The information supplied in the reigster summary table is:

Column Name	Meaning
Register Name	The symbolic name of the register
HW Field	The bits of the register as implemented in the hardware
SW Field	The bits of the register as used by the software
Address	The physical address of the register
Access Type	Indicates whether the register is Read Only (R), Write Only (W), or can be both Read and Written (R/W)
Prot	Indicates whether the registerr is writable from command packets in the FIFO or Clip List
Description	A brief description of the register

Table 3: LCG Registers

Register Name	HW Field	SW Field	Address	Access Type	Prot	Description
Memory Control, Flow Control, Configuration Register Addresses						
MEM_CONFIG	<15:00>	<15:00>	2010.1800	R/W	*	Contains Type & Size of Memory
MEM_STATUS	<07:00>	<07:00>	2010.1804	R	"	Memory State Machine Status
MEM_CURRENT_STATE	<03:00>	<03:00>	2010.1808	R	*	State Machine Current State
MEM_ERROR	<31:00>	<31:00>	2010.180C	R/W	"	Memory Parity Error Address
FLOW_CONTROL_STATUS	<31:00>	<31:00>	2010.1810	R	"	Flow Control State Machine Status

Table 3 (Cont.): LCG Registers

Register Name	HW Field	SW Field	Address	Access Type	Prot	Description
			Video Control Register Addresses			
VIDEO_CONFIG	<31:00>	<31:00>	2010.1E00	R/W	*	Video Configuration and Status
VIDEO_HTIMING	<29:02>	<29:00>	2010.1E10	W	*	Video Horizontal Timing Parameters
VIDEO_VTIMING	<29:00>	<29:00>	2010.1E14	W	*	Video Vertical Timing Parameters
VIDEO_TIMING	<26:02>	<26:00>	2010.1E18	R	*	Current Video Timing Parameters
VIDEO_X	<10:02>	<10:02>	2010.0E30	R		Current Video X Position
VIDEO_Y	<26:16>	<26:16>	2010.0E30	R		Current Video Y Position
VIDEO_REFRESH_BASE	<20:11>	<20:11>	2010.0E34	R		Current Video Address
VIDEO_REFRESH_SHIFT	<10:00>	<10:00>	2010.0E40	R		Pixels/4 from Video Shift Register
VIDEO_LUT_LOAD_COUNT	<28:16>	<28:16>	2010.0E40	R		Bytes Loaded Into LUT x 4
CURSOR_SCANLINE_LW0	<31:00>	<31:00>	2010.0E50	W		Cursor Scan Line Buffer Longword 0
CURSOR_SCANLINE_LW1	<31:00>	<31:00>	2010.0E54	W		Cursor Scan Line Buffer Longword 1
CURSOR_SCANLINE_LW2	<31:00>	<31:00>	2010.0E58	W		Cursor Scan Line Buffer Longword 2
CURSOR_SCANLINE_LW3	<31:00>	<31:00>	2010.0E5C	W		Cursor Scan Line Buffer Longword 3
CURSOR_BASE	<26:00>	<26:00>	2010.0E80	R/W		Cursor Pattern Address
CURSOR_XY	<31:00>	<31:00>	2010.0E84	R/W		Cursor X and Y Postion
CURSOR_X	<11:00>	<11:00>	2010.0E84	R/W		Cursor X Postion
CURSOR_Y	<27:16>	<27:16>	2010.0E84	R/W		Cursor Y Postion
LUT_CONSOLE_SEL	<00:00>	<00:00>	2010.0EE0	W		Server/Console(21000000) LUT Select
LUT_COLOR_BASE_W	<20:11>	<20:00>	2010.06E4	W		Server LUT Data Start Address
LUT_COLOR_BASE_R	<23:14>	<23:03>	2010.06E4	R		Current LUT Data Start Address
LUT_CONTROL_BASE	<20:11>	<20:00>	2010.0EE8	R/W		Control LUT Data Start Address
VIDEO_COUNTER_TEST	<04:00>	<04:00>	2010.0F00	R/W		Video Counter Diagnostic Carries
MEM_REFRESH_BASE	<23:14>	<23:14>	2010.0F04	R		Memory Refresh Address

Table 3 (Cont.): LCG Registers

Register Name	HW Field	SW Field	Address	Access Type	Prot	Description
Graphics Control and VM Register Addresses						
LCG_GO	<03:00>	<03:00>	2010.0C80	W		AG, Command FIFO, Clip List GO bits
NEXT_ADDRESS	<29:01>	<29:00>	2010.1334	R	*	Next Virtual/Physical Address
PA_SPTE_PTE	<26:02>	<26:00>	2010.1338	R	*	SPTE Which Maps the Process PTE
TB_INVALIDATE_SINGLE	<31:00>	<31:00>	2010.1A00	W		Invalidate a Single TB
TB_INVALIDATE_ALL	<00:00>	<00:00>	2010.1A08	W		Invalidate All TBs
TB_INVALIDATE_STATUS	<00:00>	<00:00>	2010.1A10	R		Invalidate ACK Bit
TB_STATUS	<31:00>	<31:00>	2010.1C00	R		Virtual Translation Status
TB_VPN_COUNT	<15:00>	<15:00>	2010.1C04	R		Number of TB VPNs
TB_DEST_VPN	<29:09>	<29:09>	2010.1C14	R		Destination Virtual Page Number
TB_SOURCE_VPN	<29:09>	<29:09>	2010.1C18	R		Source Virtual Page Number
TB_STENCIL_VPN	<29:09>	<29:09>	2010.1C1C	R		Stencil Virtual Page Number
TB_DEST_DATA_PFN_R	<26:09>	<29:00>	2010.1C24	R	*	Destination Physical Frame Number
TB_DEST_DATA_PFN_W	<17:00>	<17:00>	2010.1C24	W	*	Destination Physical Frame Number
TB_SOURCE_DATA_PFN_R	<26:09>	<29:00>	2010.1C28	R	*	Source Physical Frame Number
TB_SOURCE_DATA_PFN_W	<17:00>	<17:00>	2010.1C28	W	*	Source Physical Frame Number
TB_STENCIL_DATA_PFN_R	<26:09>	<29:00>	2010.1C2C	R	*	Stencil Physical Frame Number
TB_STENCIL_DATA_PFN_W	<17:00>	<17:00>	2010.1C2C	W	*	Stencil Physical Frame Number
TB_DEST_PTE_PFN_R	<26:09>	<26:00>	2010.1C34	R	*	Destination P0 Page Table Page
TB_DEST_PTE_PFN_W	<17:02>	<17:00>	2010.1C34	W	*	Destination P0 Page Table Page
TB_SOURCE_PTE_PFN_R	<26:09>	<26:00>	2010.1C38	R	*	Source P0 Page Table Page
TB_SOURCE_PTE_PFN_W	<17:02>	<17:00>	2010.1C38	W	*	Source P0 Page Table Page
TB_STENCIL_PTE_PFN_R	<26:09>	<26:00>	2010.1C3C	R	*	Stencil P0 Page Table Page
TB_STENCIL_PTE_PFN_W	<17:02>	<17:00>	2010.1C3C	W	*	Stencil P0 Page Table Page
GRAPHICS_CONFIG	<15:00>	<15:00>	2010.1C90	R	*	Graphics Configuration/Verion Number
GRAPHICS_INT_STATUS	<31:00>	<31:00>	2010.1C94	R/W	*	Graphics Interrupt/Error Status
GRAPHICS_INT_SET_ENABLE	<31:00>	<31:00>	2010.1C98	R/W	*	Interrupt Enables
GRAPHICS_INT_CLR_ENABLE	<31:00>	<31:00>	2010.1C9C	W	*	Interrupt Disables
GRAPHICS_SUB_STATUS	<31:00>	<31:00>	2010.1CA0	R/W	*	Graphics Subsystem Status
GRAPHICS_CONTROL	<31:08>	<31:00>	2010.1CA4	R/W	*	Graphics Subsystem Control
BREAKPT_ADDRESS	<29:09>	<29:00>	2010.1CB0	R/W	*	Address to break at
BREAKPT_VIRTUAL	<00:00>	<00:00>	2010.1CB0	R/W	*	Breakpoint Address is a virtual address
WRITE_PROTECT_LOW_HIGH	<26:00>	<26:00>	2010.1CC0	R/W	*	Lowest Writeble Physical Address
WRITE_PROTECT_LOW	<10:00>	<10:00>	2010.1CC0	R/W	*	Lowest Writeble Physical Address
WRITE_PROTECT_HIGH	<26:16>	<26:16>	2010.1CC0	R/W	*	Highest Writeble Physical Address
MAX_VIRTUAL_ADDRESS	<20:02>	<20:00>	2010.2350	W	*	Max Virtual Address
PA_SPTE_P0BR	<26:02>	<26:00>	2010.2354	R/W	*	System PTE Which Maps P0 Base

Table 3 (Cont.): LCG Registers

Register Name	HW Field	SW Field	Address	Access Type	Prot	Description
Clip List / Command FIFO Register Addresses						
CLIP_LIST_OFFSET	<15:04>	<15:00>	2010.04E4	R/W		Clip List Address
CLIP_LIST_BASE	<26:16>	<26:16>	2010.04E4	R/W		Clip List Address
CLIP_LIST	<26:04>	<26:00>	2010.04E4	R/W		Clip List Address
FIFO_MASKS	<15:00>	<31:00>	2010.0570	R/W		FIFO Length, Full, & Empty Masks
FIFO_HEAD_OFFSET	<15:04>	<15:00>	2010.0574	R/W		Command FIFO Read Address
FIFO_BASE	<26:16>	<26:16>	2010.0574	R/W		Command FIFO Read Address
FIFO_HEAD	<26:04>	<26:00>	2010.0574	R/W		Command FIFO Read Address
FIFO_TAIL_OFFSET	<15:04>	<15:00>	2010.0578	R/W		Command FIFO Write Address
FIFO_BASE2	<26:16>	<26:16>	2010.0578	R/W		Command FIFO Write Address
FIFO_TAIL	<26:04>	<26:00>	2010.0578	R/W		Command FIFO Write Address
CLIP_LIST_SAVE_OFFSET	<15:04>	<31:00>	2010.0CE4	R		Clip List Save Offset Latch
FIFO_RESIDUE_LW0	<31:00>	<31:00>	2010.0D04	R/W		Command FIFO Residue Longword 0
FIFO_RESIDUE_LW1	<31:00>	<31:00>	2010.0D08	R/W		Command FIFO Residue Longword 1
FIFO_RESIDUE_LW2	<31:00>	<31:00>	2010.0D0C	R/W		Command FIFO Residue Longword 2
FIFO_LENGTH	<13:00>	<13:00>	2010.0D70	R		Command FIFO Length
FIFO_SAVE_HEAD_OFFSET	<15:04>	<15:00>	2010.0D74	R		Command FIFO Save Head
FIFO_WINDOW_BASE	<29:00>	<29:00>	2018.0000	R		Command FIFO Window Base
FIFO_WINDOW_END	<29:00>	<29:00>	2020.0000	R		Command FIFO Window End
Graphics Data Buffer and Pixel SLU Register Addresses						
LOGICAL_FUNCTION	<03:00>	<03:00>	2010.0220	W		X Windows Boolean Function
PLANE_MASK	<07:00>	<07:00>	2010.0234	W		Plane Mask for Updating Planes
SOURCE_PLANE_INDEX	<04:00>	<04:00>	2010.026C	W		Plane Index for Compaction
FOREGROUND_PIXEL	<07:00>	<07:00>	2010.02C0	W		Foreground Pixel
BACKGROUND_PIXEL	<07:00>	<07:00>	2010.04C0	W		Background Pixel
GDB_LW0	<31:00>	<31:00>	2010.0D80	R/W		Graphics Data Buffer Longword 0
GDB_LW1	<31:00>	<31:00>	2010.0D84	R/W		Graphics Data Buffer Longword 1
GDB_LW2	<31:00>	<31:00>	2010.0D88	R/W		Graphics Data Buffer Longword 2
GDB_LW3	<31:00>	<31:00>	2010.0D8C	R/W		Graphics Data Buffer Longword 3
GDB_LW4	<31:00>	<31:00>	2010.0D90	R/W		Graphics Data Buffer Longword 4
GDB_LW5	<31:00>	<31:00>	2010.0D94	R/W		Graphics Data Buffer Longword 5
GDB_LW6	<31:00>	<31:00>	2010.0D98	R/W		Graphics Data Buffer Longword 6
GDB_LW7	<31:00>	<31:00>	2010.0D9C	R/W		Graphics Data Buffer Longword 7
SLU_STATE	<08:00>	<08:00>	2010.0DA0			Pixel SLU Status

Table 3 (Cont.): LCG Registers

Register Name	HW Field	SW Field	Address	Access Type	Prot	Description
LCG Address Generator Register Addresses						
CLIP_MIN_Y	<29:02>	<29:00>	2010.0244	R/W		Y Min LW Address for clipping
CLIP_MIN_MAX_X	<31:00>	<31:00>	2010.0248	R/W		X Min Coordinate for clipping
CLIP_MIN_X	<15:00>	<15:00>	2010.0248	R/W		X Min Coordinate for clipping
CLIP_MAX_X	<31:16>	<31:16>	2010.0248	R/W		X Max Coordinate for clipping
CLIP_MAX_Y	<29:02>	<29:00>	2010.024C	R/W		Y Max LW Address for clipping
DEST_X_BIAS	<15:00>	<15:00>	2010.0250	R/W		Destination X Bias
DEST_Y_ORIGIN	<29:02>	<29:00>	2010.0254	R/W		Destination Y Origin
DEST_Y_STEP	<15:02>	<15:00>	2010.0258	R/W		Destination Y Step
SOURCE_X_BIAS	<15:00>	<15:00>	2010.0260	R/W		Source X Bias
SOURCE_Y_BASE	<29:02>	<29:00>	2010.0264	R/W		Source Y Base
SOURCE_Y_STEP_WIDTH	<31:02>	<31:00>	2010.0268	R/W		Source Y Step
SOURCE_Y_STEP	<15:02>	<15:00>	2010.0268	R/W		Source Y Step
SOURCE_WIDTH	<31:16>	<31:16>	2010.0268	R/W		Source Width
STENCIL_X_BIAS	<15:00>	<15:00>	2010.0270	R/W		Stencil X Bias
STENCIL_Y_BASE	<29:02>	<29:00>	2010.0274	R/W		Stencil Y Base
STENCIL_Y_STEP	<15:02>	<15:00>	2010.0278	R/W		Stencil Y Step
DEST_Y_BASE	<29:02>	<29:00>	2010.0284	R/W		Destination Base
DEST_X	<15:00>	<15:00>	2010.0290	R/W		Destination X Position
DEST_WIDTH_HEIGHT	<31:00>	<31:00>	2010.0294	R/W		Destination Width
DEST_WIDTH	<15:00>	<15:00>	2010.0294	R/W		Destination Width
DEST_HEIGHT	<31:16>	<31:16>	2010.0294	R/W		Destination Height
AG_STATUS2	<31:00>	<31:00>	2010.0320	R/W		Opcode Action Code
AG_CURRENT_STATE	<04:00>	<15:00>	2010.0320	R		Address Generator's Current State
CURRENT_OPCODE	<23:16>	<23:16>	2010.0320	R		Current Command Opcode
OP_ACTION_CODE	<27:24>	<31:24>	2010.0320	R		Opcode Action Code
AG_STATUS	<15:00>	<15:00>	2010.0324	R		Address Generator Status
NEXT_X	<15:00>	<15:00>	2010.0330	R		Temporary Register for X Coordinate
CLIP_X_DIFF	<31:16>	<31:16>	2010.0330	R		Clip Difference for X Coordinate
SOURCE_X_BIAS0	<15:00>	<15:00>	2010.0460	R		Starting Source X Bias
SOURCE_WIDTH0	<31:16>	<31:16>	2010.0468	R		Starting Source Width
DEST_X0	<15:00>	<15:00>	2010.0490	R		Starting Destination X Position
DEST_WIDTH0	<15:00>	<15:00>	2010.0494	R/W		Starting Destination Width
TILE_ROTATION	<15:00>	<15:00>	2010.0660	R/W		Tile Width offset - 1st Access
TILE_WIDTH	<31:16>	<31:16>	2010.0668	R/W		Tile Width

11.4.1 LCG Address Generator Register Addresses (LCG I Implementation)

This table describes all of the addresses which access the Address Generator's interal registers. The multiple addresses are for simplification of the address and command packet register access decode. The registers rows which start with a (*) are the documented formal addresses. Addressing these registers with one of the other addresses may cause side effects which are not documented here.

	REGISTER NAME	I/O ADDRESS	SIZE
	CLIP_Y_MIN	2010 0044	L
*	CLIP_Y_MIN	2010 0244	L
	CLIP_X_MIN	2010 0048	W
*	CLIP_X_MIN	2010 0248	W
	CLIP_X_MAX	2010 004A	W
*	CLIP_X_MAX	2010 024A	W
	CLIP_Y_MAX	2010 004C	L
*	CLIP_Y_MAX	2010 024C	L
	DST_X_BIAS	2010 0050	W
*	DST_X_BIAS	2010 0250	W
	DST_Y_ORIG	2010 0054	L
*	DST_Y_ORIG	2010 0254	L
	DST_Y_STEP	2010 0058	W
*	DST_Y_STEP	2010 0258	W
	SRC_X_BIAS	2010 0060	W
*	SRC_X_BIAS	2010 0260	W
	SRC_WIDTH	2010 006A	W
*	SRC_WIDTH	2010 026A	W
	SRC_Y_STEP	2010 0068	W
	SRC_Y_STEP	2010 00BA	W
*	SRC_Y_STEP	2010 0268	W
	STENCIL_X_BIAS	2010 0070	W
*	STENCIL_X_BIAS	2010 0270	W
	STENCIL_Y_BASE	2010 0074	L
*	STENCIL_Y_BASE	2010 0274	L
	STENCIL_Y_STEP	2010 0078	W
	STENCIL_Y_STEP	2010 00B8	W
*	STENCIL_Y_STEP	2010 0278	W
	DST_Y_BASE	2010 0084	L
	DST_Y_BASE	2010 0098	L
	DST_Y_BASE	2010 00A4	L
	DST_Y_BASE	2010 00B4	L
	DST_Y_BASE	2010 00D4	L
	DST_Y_BASE	2010 0114	L
	DST_Y_BASE	2010 0194	L
*	DST_Y_BASE	2010 0284	L
	DST_X	2010 0080	W
	DST_X	2010 0090	W
	DST_X	2010 00A0	W
	DST_X	2010 00B0	W
*	DST_X	2010 0290	W

```
  DST_WIDTH          2010 0094     W
  DST_WIDTH          2010 00A8     W
  DST_WIDTH          2010 00BC     W
* DST_WIDTH          2010 0294     W

  DST_HEIGHT         2010 0096     W
  DST_HEIGHT         2010 00AA     W
  DST_HEIGHT         2010 00BC     W
* DST_HEIGHT         2010 0296     W

  SRC_Y_BASE         2010 0064     L
  SRC_Y_BASE         2010 00AC     L
* SRC_Y_BASE         2010 0264     L

* AG_CURRENT_STATE   2010 0320     B

  OP_ACT_CODE        2010 0010     B
* OP_ACT_CODE        2010 0321     B

* CURRENT_OPCODE     2010 0322     B

* OP_LW_CNT          2010 0323     B

* AG_STATUS          2010 0324     L

* NEXT_X             2010 0330     W

* X_CLIP_DIFF        2010 0332     W

* SRC_SHIFT_COUNT    2010 0358     B

* STENCIL_SHIFT_C    2010 0359     B

* DST_SHIFT_TERM     2010 035C     B

* SRC_SHIFT_TERM     2010 035D     B

* STENCIL_SHIFT_T    2010 035E     B

  SRC_X_BIAS0        2010 0060     W
* SRC_X_BIAS0        2010 0460     W

  SRC_WIDTH0         2010 006A     W
* SRC_WIDTH0         2010 046A     W

  DST_X0             2010 0080     W
  DST_X0             2010 0090     W
  DST_X0             2010 00A0     W
  DST_X0             2010 00B0     W
* DST_X0             2010 0490     W

  DST_WIDTH0         2010 0094     W
  DST_WIDTH0         2010 00A8     W
  DST_WIDTH0         2010 00BC     W
* DST_WIDTH0         2010 0494     W

  TILE_WIDTH         2010 006A     W
* TILE_WIDTH         2010 066A     W

  TILE_ROTATION      2010 0060     W
* TILE_ROTATION      2010 0660     W
```

11.5 Low Cost Graphics Register Descriptions

The following is a description of the registers within the memory, graphics, and video controller. The general flow and use of these registers is described, but for a better description reference the LCG Command Packet and Address Generator specifications, as well as, the sections of this specification which describe the particular subsystem in which the register/latch is used.

The descriptions below refer to the internal registers as either registers or latches. The distinction here is that a latch is a flow through element and is less gates than the edge triggered register. Latches are used whenever possible to save gates.

Words which are in all uppercase characters are the symbolic name for the register, bit field, or value being described. The one exception is for acronyms such as LCG, FIFO, VRAM, etc., which are also in all uppercase characters.

11.5.1 Memory Configuration Latch - MEM_CONFIG

This latch contains the current configuration of main memory.

The 256kx4 DRAMs cannot be mixed with 1Mx1s nor 4Mx1s. The 256kx4 DRAM configurations are meant for the lowest possible cost system with a maximum memory capacity of 8M bytes. The PMariah system requires the ability to mix 1Mx1s and 4Mx1s within the memory system, but this is not be supported in pass 1 of the S-Chip ASIC. The 1Mx1s and 4Mx1s cannot be mixed within a memory bank. The 2M byte banks (256kx4 DRAMs) use the same encoding as the 8M byte banks (1Mx1 DRAMs).

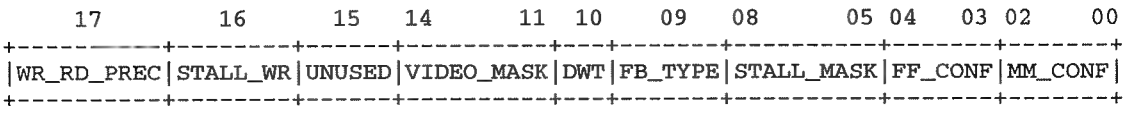


Table 4: Format of MEM_CONFIG

Name	Field	Description
MEM_STATE	<31:18>	Read Only - Assorted Memory State - TBS
MEM_WR_RD_PRECHARGE	<17:17>	Stall on cycle with write-read, between them
MEM_STALL_ON_WRITE	<16:16>	Causes a one cycle stall on Option writes
MEM_UNUSED1	<15:15>	Unused Bit
MEM_VIDEO_MASK	<14:11>	Shift register load enable for each FB bank
MEM_DWT	<10:10>	Decwindows terminal 2M Byte Memory Bank enable
MEM_FB_TYPE	<09:09>	Monochrome/color frame buffer
MEM_STALL_MASK	<08:05>	Stall on a particular FB memory bank
MEM_FB_CONFIG	<04:03>	Frame buffer memory type
MEM_MM_CONFIG	<02:00>	8 meg or 32 meg bank mask

11.5.1.1 RAS Precharge Stall Control

Setting this bit to a 0 will stall the request honoring process for back to back write - read cycles. This is to meet RAS precharge specifications for the memories used.

11.5.1.2 Write Stall Control

Setting this bit to a 1 will disable the "free" stall cycle on frame buffer writes. A 0 will enable the stall. This bit works in conjuntion with the Slow Memory Stall Mask. Reads are unaffected. The default value is 0.

11.5.1.3 Video Access Mask

These bits control the MVBUS values (RAS masks) for shift register load operations.

11.5.1.4 DWT

Setting this bit to a 1 will configure the system to be a DWT. That is, base memory of 2M and 2M increments on the banks, allowing 8M of total memory.

11.5.1.5 Monochrome

Setting this bit to 1 results in monochrome flows. The default value is 0.

11.5.1.6 Slow Frame Buffer Memory Stall Mask

Writing a 1 to these bits will stall the memory state machine and memory cycles, when frame buffer accesses are in this range. The default value is 0000.

11.5.1.7 Frame Buffer configuration

Table 5: Possible Values of MEM_FBx

Value	Description
00	1Mx1 32MB max Memory Range
01	Undefined
10	128kx8 4MB max Memory Range
11	256kx4 8B max Memory Range

11.5.1.8 System Memory Configuration

The following table shows the only possible values for the MEM_MM_CONFIG field within the Memory Configuration Latch<02:00>.

Table 6: Possible Values of MEM_BANKx

Value	Description
000	32MB max Memory Range - PVAX2 or PMariah
100	56MB max Memory Range - PMariah Only
110	80MB max Memory Range - PMariah Only
111	104MB max Memory Range - PMariah Only

11.5.2 Memory Status Latch - MEM_STATUS

TBS - SRINI

11.5.3 Memory Next State Latch - MEM_CURRENT_STATE

TBS - SRINI

11.5.4 Memory Error Information - MEM_ERROR

The PVAX2/PMariah memory subsystem provides byte parity. A parity error is the only type of memory error reported. The memory address reported is always longword aligned. This latch contains the first system wide parity error encountered. The MEM_ERR_PARITY bit will be cleared when this latch is read. The subsystem which encounters the parity error will report the error.

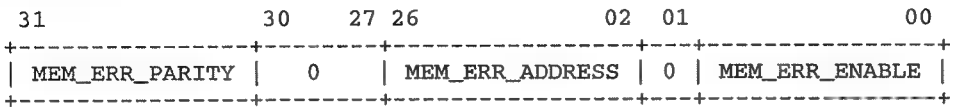


Table 7: Format of MEM_ERROR

Name	Field	Description
MEM_ERR_PARITY	<31:31>	Parity error, cleared by writing a 1
MEM_ERR_STATUS	<30:27>	Status Bits - TBS
MEM_ERR_ADDRESS	<26:02>	Address at which the parity error has occured
MEM_ERR_PARITY_TYPE	<01:01>	0 - Normal or 1 - Diagnostic Parity
MEM_ERR_ENABLE	<00:00>	Enable parity checking

11.5.5 Flow Control Status Latch - FLOW_CONTROL_STATUS

TBS - SRINI

11.5.6 Look-Up-Table Console Select Latch - LUT_CONSOLE_SEL

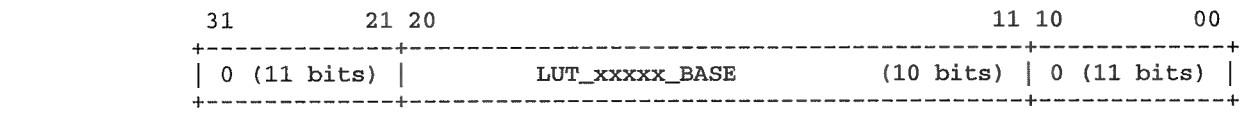
This 1 bit write only latch is used to specify whether to use the LUT_COLOR_BASE latch or the hardwired console LUT base address, 2180.000. The value of this latch can be read through the VIDEO_CONFIG latch.

Table 8: Possible Values of LUT_CONSOLE_SEL

Name	Value	Description
LUT_SEL_CONSOLE	0	Use the console LUT (2180.000)
LUT_SEL_COLOR	1	Use the LUT_COLOR_BASE

11.5.7 Color/Control LUT Base Latches - LUT_COLOR_BASE, and LUT_CONTROL_BASE

The contents of these 10 bit latches are added to physical address 2100.000 to locate the starting address of the server color LUT and control LUT data within the frame buffer. Each LUT is 2048 bytes long and must be aligned on a 1KB boundry. Both look-up-table base addresses are double buffered and are latched on every transition of active video to vertical front porch. The video shift register is used to load the LUT data during the next vertical retrace.



Whenever the LUT_CONTROL_BASE register is loaded from either the CPU or the command FIFO the VIDEO_CONTROL_LUT bit in the VIDEO_CONFIG register is automatically asserted. Once asserted, the Brooktree RAMDAC will be loaded with the control LUT data addressed by the LUT_CONTROL_BASE register and the VIDEO_CONTROL_LUT bit is cleared. On subsequent vsync's the data selected by the LUT_CONSOLE_SEL bit (either LUT_COLOR_BASE or the hardwired console LUT at 2180.000) will be loaded.

The following is an diagram of the LUT organization for the server's color palette. This format packs the data less efficiently since it does not take advantage of BT458's internal autoincrement counter. However, maintaining quadword alignment makes it easier for the server to locate and modify an arbitary color map entry.

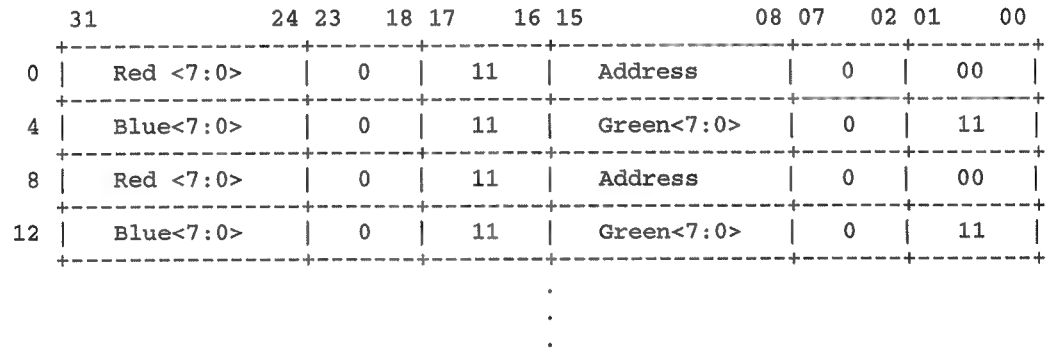
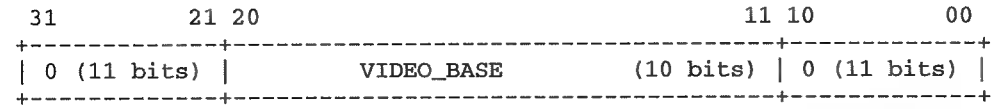


Table 9: Possible Values of LUT Data Bits <01:00>

Name	Value	Description
LUT_ADRS_REG	00	Write to LUT address register
LUT_COLOR_AUTOINC	01	Write color to LUT and autoincrement address

11.5.8 Video Base Address Latch - VIDEO_BASE

The contents of this 10 bit latch is added to physical address 2100.000 to locate the first scanline in VRAM to be displayed. This latch is loaded into the VIDEO_REFRESH_BASE at the beginning of vertical retrace.



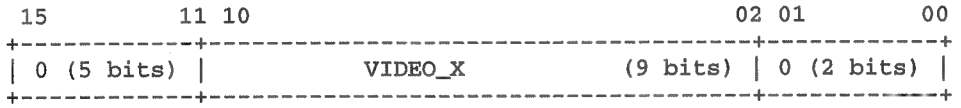
Since the video start address is aligned to the VIDEO_REFRESH_SHIFT register, what is loaded into this latch is different than the actual address. The address bits which should be loaded into this latch, for each type of video module, are shown in the following table. The least significant bit of the specified field should be put into bit<11> of the latch above. This is done in this way to avoid adding a large multiplexer at the output of the VIDEO_REFRESH_BASE to move it to the appropriate bits just so that the memory addressing logic can again multiplex it to the RAS address lines for the video memory.

Table 10: Alignments for VIDEO_BASE Values

Name	Field	VRAM Type	Description
VIDEO_8H	<20:12>	256Kx4	High resolution 8 planes
VIDEO_8L	<19:11>	128Kx8	Low resolution 8 planes
VIDEO_32_1	<18:10>	128Kx8	32 bit 1 plane
VIDEO_16_1	<17:09>	128Kx8	16 bit 1 plane

11.5.9 Video X Position Register - VIDEO_X

This 9 bit register contains the current X position of the video state machine. The X counter runs on the video clock divided by 4 and therefore requires (11 - 2 =) 9 bits which gives a range of up to 2048 pixels. This counter is used for the video timing, but not comparison for, the cursor X position.

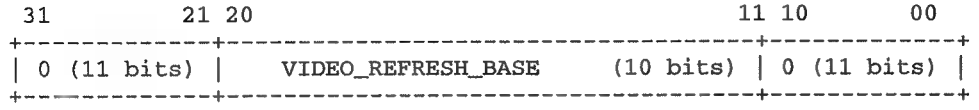


11.5.10 Video Y Position Register - VIDEO_Y

This 11 bit register contains the current Y position of the video state machine which gives a range of up to 2048 scanlines. It is updated on every scanline update. This counter is used for the video timing, but not comparison for, the cursor Y position.

11.5.11 Video Refresh Address Counter - VIDEO_REFRESH_BASE

Video addressing is performed on linear memory addresses and the address counters cannot be used for video timing nor cursor position detection. The contents of this 10 bit latch is added to physical address 2100.000 to locate the address of the VRAM row to be loaded into the video shift register (VIDEO_REFRESH_SHIFT) for display. It is reloaded with the VIDEO_BASE at the beginning of video retrace. This counter is incremented whenever the VIDEO_REFRESH_SHIFT register selected bit goes from 1 to 0.



11.5.12 Video Refresh Counter - VIDEO_REFRESH_SHIFT

This counter keeps track of the number of pixels (divided by 4) used in the video shift register. The VIDEO_CONFIG latch contains the bits for selecting the bit number for incrementing the VIDEO_REFRESH_BASE and making the video shift register load request. The table below indicates the number of pixels between shift register loads for the supported video modules.

Table 11: Pixels Between Shift Register Loads

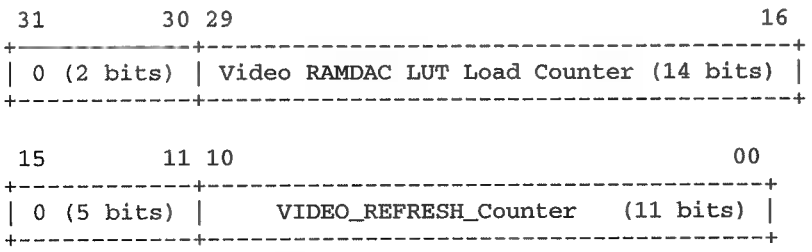
Resolution	1 Plane	8 Plane	Type of Load
800 x 600	2048	1024	Split Load
1024 x 768	4096	2048	Full Load
1280 x 1024	2048	2048	Split Load
1536 x 1152	2048	2048	Split Load

Table 11 (Cont.): Pixels Between Shift Register Loads

Resolution	1 Plane	▪ Plane	Type of Load
2048 x 1536	8192	4096	Full Load

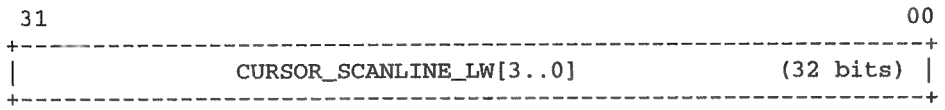
This counter is also used for memory refresh. A separate set of select bits in the VIDEO_CONFIG latch are used for selecting the bit number of this register for incementing the MEM_REFRESH_BASE and making the memory refresh request.

The RAMDAC LUT Load Counter is also read back with this address on the most significant word. Bits <29:28,20:16> are asserted high, while bits <27:21> are inverted. Reading back this counter is meant only for test coverage.



11.5.13 Cursor Scanline Buffer Latches - CURSOR_SCANLINE_LW[3..0]

Each of these four 32 bit latches buffer ▪ scanline of the 64x64x2 cursor for both the 8 plane color and the monochrome video subsystems. All 4 latches are loaded during horizontal retrace with an octaword read which is requested on the horizontal front porch to sync transition of active cursor scanlines.



11.5.14 Cursor Base Address Latch - CURSOR_BASE

This double buffered latch contains the address of the cursor pattern data. The 1024 bytes of cursor pattern data can be stored anywhere in main memory, but must be 1024 byte aligned and be physically contiguous. The cursor cannot be stored in the frame buffer. This is because the cursor pattern data is read a scanline (octaword) at a time during horizontal retrace (horizontal sync and back porch) of scanlines where the cursor is active. The monochrome frame buffer is not capable of octaword acceeses.

Only bits <26:10> and bit <0> can be written. Reading bits <09:04> return the current cursor scanline number, which should normally be zero unless the read takes places in the middle of a cursor update. Reading bits <26:04> return the current octaword address of the cursor address latch. Bits <26:10> of this latch are double buffered. The active address latch is loaded on the active video to vertical front porch transition.

The CURSOR_ENABLE bit enables display of the cursor when written with a zero. When display of the cursor is disabled, cursor memory reads are also disabled. This bit can also be set or disabled by modifying the VIDEO_CURSOR bit the the VIDEO_CONFIG register. It is suggested that cursor display be disabled by clearing the VIDEO_CURSOR

bit in VIDEO_CONFIG, and that the cursor display be enabled by writing a 1k byte aligned address to the CURSOR_BASE register.

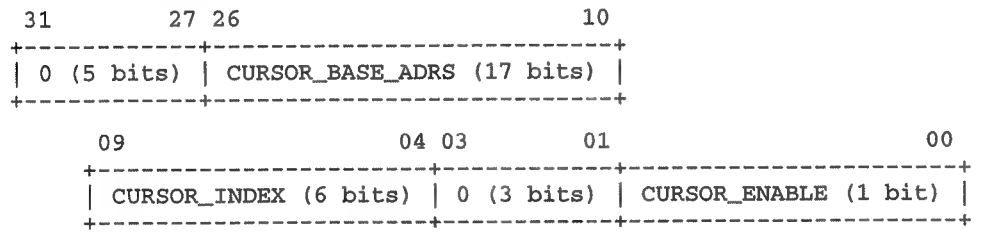


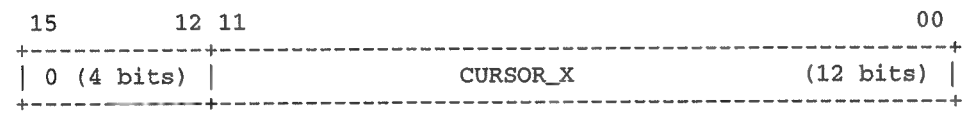
Table 12: Format of CURSOR_BASE

Name	Field	Description
CURSOR_UNUSED0	<31:27>	Unused bits
CURSOR_BASE_ADRS	<26:10>	Base address
CURSOR_INDEX	<09:04>	Cursor scanline index
CURSOR_UNUSED1	<03:01>	Unused bits
CURSOR_ENABLE	<00:00>	Cursor display enable

11.5.15 Cursor X Position Latch - CURSOR_X

This double buffered latch is used for positioning the left most pixel, as seen on the screen, of the cursor. This latch is double buffered. The active comparison latch is loaded on the active video to vertical front porch transition.

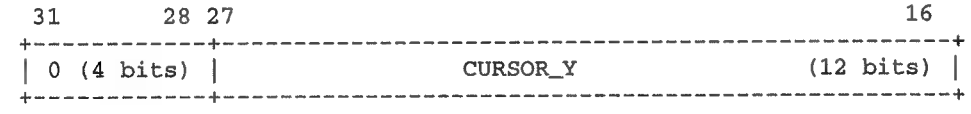
The horizontal sync and back porch timing parameters must be added together and added to the CURSOR_X to position the cursor on the screen during active video. The cursor positon detection logic is implemented in this way so that the cursor can be positioned either totally or partially off the visible video.



11.5.16 Cursor Y Position Latch - CURSOR_Y

This double buffered latch is used positioning the first (top) scanline of the cursor. This latch is double buffered. The active comparison latch is loaded on the active video to vertical front porch transition.

The vertical sync and back porch timing parameters must be added together and added to the CURSOR_Y to position the cursor on the screen during active video. The cursor positon detection logic is implemented in this way so that the cursor can be positioned either totally or partially off the visible video.



11.5.17 Video Configuration Latch - VIDEO_CONFIG

This latch contains the video configuration and control information. The following list describes the state of this latch after system reset.

- Cursor is disabled
- Cursor Pins are configured to MV_DAL Option
- Video timing and Memory refresh is disabled
- Video is Blanked
- Sync is Disabled
- Video clock select = CPU_CLOCK
- horizontal and vertical state is set to front porch
- horizontal and vertical pixel and scan line counters are cleared
- horizontal and vertical timing parameters are not effected
- all active registers are written from the load registers

The following table describes the format of this register.

Table 13: Format of VIDEO_CONFIG

Name	Field	Description
VIDEO_VSTATE	<31:30>	Vertical State
VIDEO_HSTATE	<29:28>	Horizontal State
VIDEO_UNUSED0	<27:26>	Unused Bits
VIDEO_CONSOLE_LUT	<25:25>	Console LUT Select
VIDEO_CONTROL_LUT	<24:24>	Control LUT Select
VIDEO_UNUSED1	<23:23>	Unused Bit
VIDEO_CURSOR_ACTIVE	<22:22>	Cursor Active
VIDEO_CURSOR_SCANLINE	<21:16>	Scanline of the cursor we are showing
VIDEO_RESET	<15:15>	Video Reset
VIDEO_UNUSED2	<14:14>	Unused Bit
VIDEO_LUT_LOAD_SIZE	<13:13>	Size of LUT to load: 0 = 1k, 1 = 2k
VIDEO_UNUSED3	<12:12>	Video Sync Enable H
VIDEO_LUT_SHIFT_SEL	<11:11>	Full/split load for LUTs
VIDEO_CLOCK_SEL	<10:10>	Video Clock Select
VIDEO_MEM_REFRESH_SEL	<09:08>	Memory Refresh Select
VIDEO_REFRESH_SEL	<07:06>	Video Refresh Select
VIDEO_SHIFT_SEL	<05:05>	1 = Full load or 0 = Split load
VIDEO_CURSOR_PIN_TYPE	<04:04>	1 = real cursor output, 0 = Option control bits
VIDEO_LUT_LOAD_ENABLE	<03:03>	LUT load enable
VIDEO_CURSOR_ENABLE	<02:02>	Enable the cursor
VIDEO_ENABLE_VIDEO	<01:01>	Video Enable, otherwise screen is blanked
VIDEO_TIMING_ENABLE	<00:00>	Clock Timing Enable for video and memory refresh

11.5.17.1 Vertical State - VIDEO_VSTATE

This field is the vertical state counter's contents. Refer to the video subsystem section for details of the video timing. Possible values of this field are:

Table 14: Possible Values of VIDEO_VSTATE

Name	Value	Description
VIDEO_VFRONT_PORCH	00	Vertical Front Porch
VIDEO_VSYNC	01	Vertical Front Porch
VIDEO_VBACK_PORCH	10	Vertical Back Porch
VIDEO_VACTIVE	11	Vertical Active Video

11.5.17.2 Horizontal State - VIDEO_HSTATE

This field is the horizontal state counter's contents. Refer to the video subsystem section details of the video timing. Possible values of this field are:

Table 15: Possible Values of VIDEO_HSTATE

Name	Value	Description
VIDEO_HFRONT_PORCH	00	Horizontal Front Porch
VIDEO_HSYNC	01	Horizontal Front Porch
VIDEO_HBACK_PORCH	10	Horizontal Back Porch
VIDEO_HACTIVE	11	Horizontal Active Video

11.5.17.3 LUT Console Select - VIDEO_CONSOLE_LUT

This bit indicates whether the console LUT address 2180.000 (when 0) or the server color LUT address LUT_COLOR_BASE (when 1) is used for loading the color LUT. This bit is read only and can be written through the LUT_CONSOLE_SEL latch.

11.5.17.4 LUT Control Select - VIDEO_CONTROL_LUT

This bit indicates whether the control LUT address LUT_CONTROL_BASE (when 1) or the color LUT address selected by VIDEO_CONSOLE_LUT (when 0) is used on the next LUT load. This bit is asserted by internal control whenever the LUT_CONTROL_BASE latch is loaded. When this bit is asserted the Brooktree DAC will be loaded with the contents of the frame buffer off-screen memory addressed by the LUT_CONTROL_BASE address latch during the next vertical retrace. The LUT_CONSOLE_SEL bit will be ignored and this bit will be cleared after the video shift register load.

11.5.17.5 Cursor Active Scanline - VIDEO_CURSOR_ACTIVE

This bit is asserted on active video scanlines in which display of the cursor is present. This bit is read only and is asserted by the cursor position detection logic.

11.5.17.6 Cursor Scanline - VIDEO_CURSOR_SCANLINE

This field contains the next scanline number of the cursor to be displayed. This value is controled by the cursor position detection logic.

11.5.17.7 Reset Video - VIDEO_RESET

When this bit is written with a 1, the video subsystem will be reset to its initial state as described above. This bit is write only.

11.5.17.8 Video LUT Load Size - VIDEO_LUT_LOAD_SIZE

This bit controls the number of bytes loaded into the RAMDAC. When this bit is a (0), then 1k bytes are loaded. When this bit is a (1), then 2k bytes are loaded.

11.5.17.9 Video Sync Enable - VIDEO_SYNC_ENABLE

This bit enables the Video Sync Signal when written with a (1).

11.5.17.10 Video LUT Shift Select - VIDEO_LUT_SHIFT_SEL

When this bit is a (1), then the least significant bit of the address is directed to the least significant bit of the VRAM RAS address for full shift register loads. When this bit is a (0), then the least significant address bit is directed to the most significant address bit of the VRAM CAS address with the rest of the address shifted down 1 bit. This allows the LUT data to be loaded into the lower half of the VRAM shift register and only 1/2 of the shift register is used for loading the RAMDAC.

11.5.17.11 Video Clock Select - VIDEO_CLOCK

This bit selects the clock to be used for running the video and memory refresh logic. Either the video nibble clock pin (when 1) or the CPU clock (when 0) can be selected.

11.5.17.12 Memory Refresh Select - VIDEO_MEM_REFRESH

This field selects the number of active video pixels to be counted between memory refreshes. This timing is adjustable for the different possible monitor timings. More detail can be found in the MEM_REFRESH_BASE description. A memory refresh occurs on every scan line during Vertical Retrace. Possible values of this field are:

Table 16: Possible Values of VIDEO_MEM_REFRESH

Name	Value	Description
MEM_REFRESH_64x4	00	256 active pixels
MEM_REFRESH_128x4	01	512 active pixels
MEM_REFRESH_256x4	10	1024 active pixels
MEM_REFRESH_512x4	11	2048 active pixels

11.5.17.13 Video Refresh Select - VIDEO_REFRESH

This field selects the number of active video pixels to be counted between video shift register loads. This is ajustable for the different possible monitor resolutions. Possible values of this field are:

Table 17: Possible Values of VIDEO_REFRESH

Name	Value	Description
VIDEO_REFRESH_256x4	00	1024 active pixels
VIDEO_REFRESH_512x4	01	2048 active pixels
VIDEO_REFRESH_1024x4	10	4096 active pixels
VIDEO_REFRESH_2048x4	11	8192 active pixels

11.5.17.14 Video Shift Register Load Select - VIDEO_SHIFT

This bit selects between full shift register loads (when 1) and half (or split) shift register loads (when 0). Full shift register loads can only be used for power of 2 (512, 1024, 2048,...) scanline monitor resolutions.

11.5.17.15 Video Cursor Pin Type - VIDEO_CURSOR_PIN_TYPE

When this bit is a (1), then all cursor pins are output enabled and ready for normal cursor operation. When this bit is a (0), then the cursor pins are used as the Option interface signals.

11.5.17.16 LUT Load Enable - VIDEO_LUT_LOAD_ENABLE

When this bit is a (1), then the RAMDAC LUT loads are enabled. When it is a (0), then LUT loads are disabled.

11.5.17.17 Cursor Enable - VIDEO_CURSOR

This bit when set enables display of the cursor. When display of the cursor is disabled cursor memory reads are also disabled. This bit can also be set by clearing the CURSOR_ENABLE bit in the CURSOR_BASE latch.

11.5.17.18 Video Blank Enable - VIDEO_BLANK

This bit enables the display of the active video. When cleared the video is blanked and no display refresh memory accesses will occur. The video timing will continue with the sync signal asserted at the appropriate time. When VIDEO_BLANK is not set, the cursor pins are forced to 0.

11.5.17.19 Video Timing Enable - VIDEO_TIMING

This bit enables the video timing when written with a 1. The video memory display refresh accesses are dependent on the VIDEO_BLANK bit.

11.5.18 Video Horizontal Timing Parameter Latch - VIDEO_HTIMING

This latch contains all of the horizontal timing parameters required for the video state machine. This latch must be written from the contents of the video module ROM. The details of the video timing can be found in the video subsystem section of this specification.

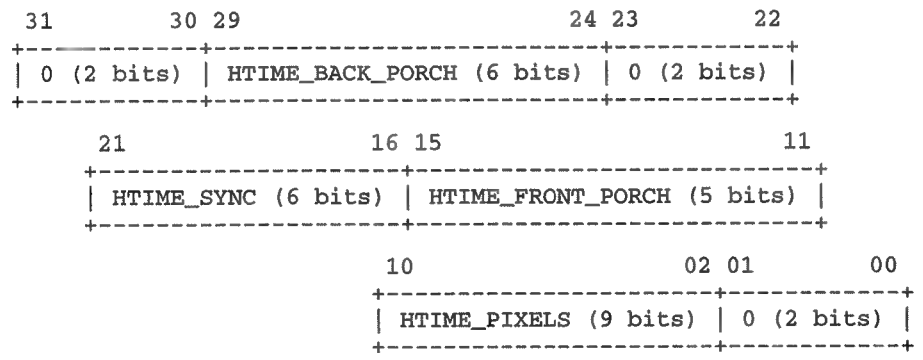


Table 18: Format of VIDEO_HTIMING

Name	Field	Description
HTIME_UNUSED0	<31:30>	Unused Bits
HTIME_BACK_PORCH	<29:24>	Horizontal Back Porch
HTIME_UNUSED1	<23:22>	Unused Bits
HTIME_SYNC	<21:16>	Horizontal Sync
HTIME_FRONT_PORCH	<15:11>	Horizontal Front Porch
HTIME_PIXELS	<10:02>	Active Pixels Per Scanline / 4
HTIME_UNUSED2	<01:00>	Unused Bits

11.5.19 Video Vertical Timing Parameter Latch - VIDEO_VTIMING

This latch contains all of the vertical timing parameters required for the video state machine. This latch must be written from the contents of the video module ROM. The details of the video timing can be found in the video subsystem section of this specification.

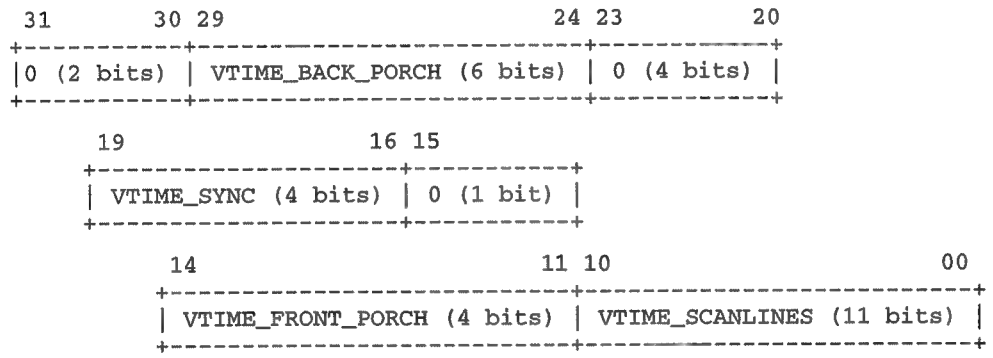


Table 19: Format of VIDEO_VTIMING

Name	Field	Description
VTIME_UNUSED0	<31:30>	Unused Bits
VTIME_BACK_PORCH	<29:24>	Vertical Back Porch
VTIME_UNUSED1	<23:20>	Unused Bits
VTIME_SYNC	<19:16>	Vertical Sync
VTIME_UNUSED2	<15:15>	Unused Bit
VTIME_FRONT_PORCH	<14:11>	Vertical Front Porch
VTIME_SCANLINES	<10:00>	Active Scanlines

11.5.20 Current Video Timing Parameters - VIDEO_TIME

This latch contains the current vertical and horizontal timing parameters which are selected by the VIDEO_VSTATE and VIDEO_HSTATE fields of the VIDEO_CONFIG latch.

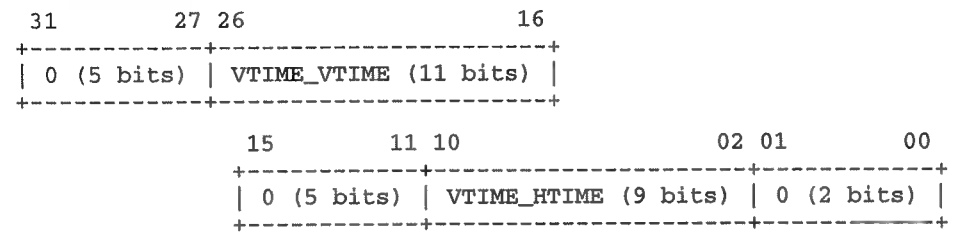


Table 20: Format of VIDEO_TIME

Name	Field	Description
VTIME_UNUSED0	<31:27>	Unused Bits
VTIME_VTIME	<26:16>	Vertical Timing
VTIME_UNUSED1	<15:11>	Unused Bits
VTIME_HTIME	<10:02>	Horizontal Timing
VTIME_UNUSED2	<01:00>	Unused Bits

11.5.21 Memory Refresh Counter - MEM_REFRESH_BASE

This 11 bit counter is used for the memory refresh address. Both main memory and video memory is refreshed at the same time. The VIDEO_REFRESH_BASE which is clocked during active video is used for generating the memory refresh accesses and incrementing the MEM_REFRESH_BASE. The VIDEO_CONFIG latch contains the select bits which determine how often memory refresh accesses occur. One memory refresh occurs per scan line during Vertical Retrace. The following tables describe the memory refresh circuitry which uses the video timing generator.

Bit Position	Video Configuration/Status Bits
09:08	Memory Refresh Select
0 0	64 x 4 = 256 pixels
0 1	128 x 4 = 512 pixels
1 0	256 x 4 = 1024 pixels
1 1	512 x 4 = 2048 pixels

Memory Refresh = 512 accesses every 8.00ms for 1M bit DRAMs
1024 accesses every 16.00ms for 4M bit DRAMs

Average Refresh = 15.625us

(Total Active Video Time)/(Total Active Pixels) = (Time/Pixel)

(15.625us)/(Time/Pixel) = ~(# Pixels\Mem Ref)

Timing Parameter	60 Hz	66 Hz	72 Hz	
Nibble Clock:				
1536 x 1152	N.A.	27.325ns	25.000ns	
1280 x 1024	N.A.	33.377ns	30.579ns	
1024 x 768	62.500ns	60.377ns	55.246ns	
800 x 600	89.087ns	N.A.	N.A.	
Vertical Active Video Period:				Pixels:
1536 x 1152	N.A.	14.6063ms	13.3632ms	1,769,472
1280 x 1024	N.A.	14.4916ms	12.9655ms	1,310,720
1024 x 768	15.7440ms	14.5601ms	13.3226ms	768,432
800 x 600	15.7149ms	N.A.	N.A.	480,000
Vertical Retrace Period:				Scan Lines:
1536 x 1152	N.A.	0.5452ms	0.4988ms	43 43
1280 x 1024	N.A.	0.5519ms	0.5056ms	39 39
1024 x 768	0.9225ms	0.5688ms	0.5204ms	45 30 30
800 x 600	1.1524ms	N.A.	N.A.	44
Active Pixels / Memory Reference:				
1536 x 1152	N.A.	1893=1024	2069=2048	
1280 x 1024	N.A.	1413=1024	1580=1024	
1024 x 768	762= 512	824= 512	901= 512	
800 x 600	477= 256	N.A.	N.A.	
Mem Ref Period:				
1536 x 1152	N.A.	8.453us	15.467us	
1280 x 1024	N.A.	11.322us	10.129us	
1024 x 768	10.490us	9.701us	8.877us	
800 x 600	8.381us	N.A.	N.A.	
Total Refresh Period:				
1536 x 1152	N.A.	4.328ms	7.919ms	
1280 x 1024	N.A.	5.797ms	5.186ms	
1024 x 768	5.371ms	4.967ms	4.550ms	
800 x 600	4.291ms	N.A.	N.A.	

When no video module is present, the memory refresh logic uses the horizontal video timing and uses the CLK_P3 signal. The video timing and memory refresh counters can be set up to achieve the optimum memory refresh time.

11.5.22 LCG Go Register - LCG_GO

This register contains the go bits which allow the LCG to be restarted after certain events have caused it to stop. Asserting the appropriate bit will cause the LCG to continue if it was stopped for any of the reasons listed. If multiple events cause the LCG to stop (eg: a translation not valid and an address breakpoint) all the appropriate go bits must be written before the LCG will continue. Clearing any bit position will have no effect. This is a write only register.

Table 21: Format of LCG_GO

Name	Field	Description
GO_UNUSED0	<31:04>	Unused Bits
GO_VM	<03:03>	Virtual Translation Go
GO_UNUSED1	<02:02>	Unused Bits
GO_AG	<01:01>	Address Generator Go
GO_FIFO	<00:00>	FIFO/Halt Go

11.5.23 Graphics Configuration Latch - GRAPHICS_CONFIG

This location returns the LCG version number. It is read only. The exact meaning of this location will be defined after pass 1 of the S-Chip ASIC. For pass 1 of the S-Chip this latch will contain a 0.

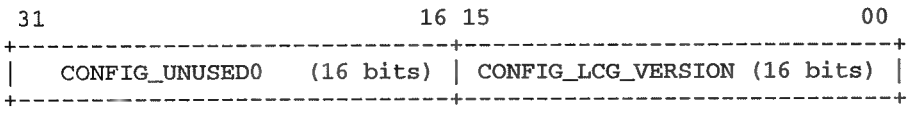


Table 22: Format of GRAPHICS_CONFIG

Name	Field	Description
CONFIG_UNUSED0	<31:16>	Unused Bits
CONFIG_LCG_VERSION	<15:00>	LCG Version Number

11.5.24 Graphics Interrupt Status Register - GRAPHICS_INT_STATUS

The bits in this register are status values set by the hardware. Asserting any of these bits will cause those status bits to be cleared. If the hardware event which caused a given status bit to be set is still present, the bit will immediately be set again by the hardware. Clearing status bits before the underlying condition has been cleared can, under certain conditions, cause spurrious interrupts to occur.

All bits in the GRAPHICS_INT_STATUS register will be forced low by system reset as well as by asserting the CTRL_RESET bit in the GRAPHICS_CONTROL register.

It is important to keep in mind that the GRAPHICS_INT_STATUS bits are flags which indicate that the specific event has occurred sometime since the bit was last cleared, and may not reflect the current state of the hardware.

Table 23: Format of GRAPHICS_INT_STATUS

Name	Field	Description
INT_VM_FAULT	<31:31>	Virtual Translation Not Valid
INT_OPTION_RESET	<30:30>	MV_DAL Reset
INT_OPTION_INHIBIT	<29:29>	MV_DAL Inhibit
INT_OPTION_STALL	<28:28>	MV_DAL Stall
INT_OPTION_INHIBIT_TO	<27:24>	MV_DAL Inhibit Stall Time Out<3:0>
INT_OPTION_STALL_TO	<23:20>	MV_DAL Option Stall Time Out<3:0>
INT_OPTION_INTERRUPT	<19:19>	MV_DAL Option Interrupt
INT_OPTION_TIMEOUT	<18:18>	MV_Dal Option Time Out
INT_FIFO_BAD_TOGGLE	<17:17>	Illegal Toggle or Load FIFO in FIFO
INT_CURSOR_PARITY	<16:16>	Cursor Parity
INT_MEM_ERROR	<15:15>	LCG Memory Error
INT_RANGE_ERROR	<14:14>	Write Range Error
INT_BAD_OPCODE	<13:13>	Unsupported Opcode Error
INT_UNUSED0	<12:12>	Unused Bit
INT_AG_ACCESS_BPT	<11:11>	AG Access Breakpoint
INT_UNUSED1	<10:10>	Unused Bit
INT_PACKET_BPT	<09:09>	LCG Command Packet Breakpoint
INT_ADDRESS_BPT	<08:08>	Address Breakpoint
INT_CLIP_LIST_WRAP	<07:07>	Clip List Wrap Error
INT_FIFO_FULL	<06:06>	Command FIFO Full
INT_FIFO_EMPTY	<05:05>	Command FIFO Empty
INT_FIFO_AFULL	<04:04>	Command Almost FIFO Full
INT_FIFO_AEMPTY	<03:03>	Command Almost FIFO Empty
INT_HALT	<02:02>	Halt Interrupt
INT_NOP	<01:01>	NOP/STORE_LONG Interrupt
INT_VSYNC	<00:00>	Video Vertical Retrace

11.5.24.1 Virtual Address Translation Not Valid - INT_VM_FAULT

This bit indicates that the Address Generator is unable to complete the current virtual to physical translation and has stalled. The GO_VM bit in the LCG_GO register must be asserted to restart the Address Generator after the offending page of memory has been made available. The NEXT_ADDRESS and TB_STATUS registers are used to locate the offending address.

11.5.24.2 Imaging/Graphics Option Interrupt - INT_GRAPHICS_OPTION

TBD/TBS.

11.5.24.3 FIFO Toggle Error - INT_FIFO_BAD_TOGGLE

This bit indicates that a bad toggle was performed in the command stream. The LCG is halted when this event occurs.

11.5.24.4 LCG Cursor Parity Error - INT_CURSOR_PARITY

This bit indicates that a parity error has occured while accessing the cursor pattern. The longword address at which the error occured is contained in the MEM_ERROR latch. The LCG is halted when this event occurs.

11.5.24.5 LCG Memory Parity Error - INT_MEM_ERROR

This bit indicates that the Address Generator, command FIFO, or clip list encountered a parity error during a main memory read access. The longword address at which the error occured is contained in the MEM_ERROR latch. The LCG is halted when this event occurs.

11.5.24.6 Write Range Error - INT_RANGE_ERROR

This bit indicates that either the command FIFO or the Address Generator has tried to write to a physical address outside of the allowed physical address range as specified by the WRITE_PROTECT_LOW and WRITE_PROTECT_HIGH registers. The LCG is halted and the write does not occur.

11.5.24.7 Unsupported OPCODE - INT_BAD_OPCODE

This bit indicates that the command FIFO or clip list encountered a command with an unsupported OPCODE. The LCG is halted when this event occurs.

11.5.24.8 Address Generator State Breakpoint - INT_AG_STATE_BPT

This bit indicates that the Address Generator state matched the value in the BREAKPT_AG_STATE register and a Address Generator state breakpoint has occured. In order for the Address Generator state breakpoint to occur it must first be enabled by setting the CTRL_AG_STATE_BPT_ARM bit in the GRAPHICS_CONTROL register. When a state breakpoint occurs the Address Generator suspends intself by setting itself to a state which is equal to the normal next state plus 32. The GO_AG_STATE bit in the LCG_GO register must be asserted to restart the Address Generator after the state breakpoint occurs.

11.5.24.9 Address Generator Access Single Step - INT_AG_ACCESS_BPT

This bit indicates that an Address Generator state machine memory access single step breakpoint has occured. In order for the Address Generator access single step breakpoint to occur it must first be enabled by setting the CTRL_AG_ACCESS_BPT_ARM bit in the GRAPHICS_CONTROL register. Once the access single step breakpoint is enabled a breakpoint occures whenever the Address Generator attempts to access memory. The Address Generator is stalled in its attempt to access memory. The GO_AG_ACCESS bit in the LCG_GO register must be asserted to restart the Address Generator after the access single step breakpoint occurs.

11.5.24.10 LCG Command Packet Single Step - INT_PACKET_BPT

This bit indicates that a command packet single step breakpoint has occured. In order for the command packet single step breakpoint to occur it must first be enabled by setting the CTRL_PACKET_BPT_ARM bit in the GRAPHICS_CONTROL register. Once the command packet single step breakpoint has been enable a breakpoint occurs whenever a command packet is unloaded from the command FIFO or clip list. The breakpoint causes the FIFO control state machine to be is stopped, and no further command packets are unloaded. The

GO_FIFO bit in the LCG_GO register must be asserted to restart the FIFO control state machine after a command packet single step breakpoint occurs.

11.5.24.11 Address Breakpoint - INT_ADDRESS_BPT

This bit indicates that the Address Generator generated and attempted to access an address whose bits <29:09> match the value in the BREAKPT_ADDRESS register and an address breakpoint has occured. In order for the address breakpoint to occur it must first be enabled by setting the CTRL_ADDRESS_BPT_ARM bit in the GRAPHICS_CONTROL register. The Address Generator is stalled in its attempt to access the address which caused the breakpoint. The GO_AG_ACCESS bit in the LCG_GO register must be asserted to restart the Address Generator after the address breakpoint occurs.

11.5.24.12 Clip List Wrap Error - INT_FIFO_CLIP_WRAP

This bit indicates that the clip list has wrapped on the 64k byte boundary limitation. The state machine interprets this event as an error condition. The LCG is halted when this event occurs.

11.5.24.13 Command FIFO Full Error - INT_FIFO_FULL

This bit indicates that an attempt was made to write a longword to the command FIFO when it was full. The LCG is halted when this event occurs, and the write to command FIFO has unpredictable results.

11.5.24.14 Command FIFO Empty - INT_FIFO_EMPTY

This bit indicates that the command FIFO is empty. This bit is latched, so by the time the interrupt is delivered to the CPU the FIFO may no longer be empty. This bit is only set when the command FIFO is completely empty. Because the FIFO control state machine performs octaword reads from the FIFO it is possible for one or more longwords to be "stuck" in the FIFO. If less that an octaword of data remains in the FIFO, the INT_FIFO_EMPTY bit will not be set. The remaining data in the FIFO may be flushed out by writing enought data to the FIFO to result in a complete octaword of data.

11.5.24.15 Command FIFO Almost Full - INT_FIFO_AFULL

This bit indicates that the command FIFO almost full threshold has been crossed. The almost full threshold may be adjusted by writing the appropriate value to the FIFO_MASKS latch.

11.5.24.16 Command FIFO Almost Empty - INT_FIFO_AEMPTY

This bit indicates that the command FIFO almost empty threshold has been crossed. The almost empty threshold may be adjusted by writing the appropriate value to the FIFO_MASKS latch.

11.5.24.17 Halt Interrupt - INT_HALT

This bit indicates that a NOP command packet has been unloaded out of the command FIFO or clip list with the HALT flag set. The GO_FIFO bit must be asserted to restart the FIFO control state machine.

Manually stopping the LCG by disabling the appropriate bits in GRAPHICS_CONTROL register will not assert this bit.

11.5.24.18 NOP/STORE_LONG Interrupt - INT_NOP

This bit indicates that a command packet has been unloaded from the command FIFO or clip list with the interrupt flag set. The status bit is set after the NOP/STORE_LONG operation has completed.

11.5.24.19 Video Vertical Retrace - INT_VSYNC

This bit indicates that the beginning of vertical retrace has occured.

11.5.25 Graphics Interrupt Enable Latch - GRAPHICS_INT_ENABLE

This latch contains the graphics hardware interrupt enables mask. This latch is implemented as two latches one which is used to set bits in this latch called GRAPHICS_INT_SET_ENABLE and one which is used to clear bits in this latch called GRAPHICS_INT_CLR_ENABLE. Asserting any of the bits the GRAPHICS_INT_SET_ENABLE register will cause the respective bits in GRAPHICS_INT_ENABLE to be set. Asserting any of the bits the GRAPHICS_INT_CLR_ENABLE
register will cause the respective bits in GRAPHICS_INT_ENABLE to be cleared. Reading either GRAPHICS_INT_SET_ENABLE or GRAPHICS_INT_CLR_ENABLE will return the current state of the GRAPHICS_INT_ENABLE. All bits will be forced to zero by system reset as well as by asserting the CTRL_RESET bit in the GRAPHICS_CONTROL register.

An interrupt is delivered to the CPU on the rising edge of the logical OR of all the bits contained in the the bitwise AND of the GRAPHICS_INT_STATUS and GRAPHICS_INT_ENABLE registers.

The organization of the bits in this register exactly match those in the GRAPHICS_INT_STATUS register. See the section which describes the GRAPHICS_INT_STATUS register for a complete description of all the bits.

11.5.26 Graphics Subsystem Status Register - GRAPHICS_SUB_STATUS

This unlatched register contains the current state of many internal LCG signals. It may be polled to determine the current state of the LCG hardware. These bits are cleared on system reset or by asserting the CTRL_RESET bit the GRAPHICS_CONTROL register.

Table 24: Format of GRAPHICS_SUB_STATUS

Name	Field	Description
GSS_AG_BUSY	<31:31>	AG Busy
GSS_UNUSED0	<30:30>	Unused Bit
GSS_SHORT_CIRCUIT	<29:29>	In Short Circuit Mode
GSS_VALID_PACKET	<28:28>	Valid Command Packet
GSS_1ST_LONGWORD	<27:27>	1st Longword Of Packet
GSS_FIFO_ARBITRATE	<26:26>	Command FIFO Arbitrate
GSS_FIFO_COMMANDER	<25:24>	Command FIFO Commander Id
GSS_AG_BACKWARDS	<23:23>	AG Backwards Access
GSS_AG_VIRTUAL	<22:22>	AG Virtual Access
GSS_AG_ARBITRATE	<21:21>	AG Arbitrate
GSS_AG_READ_ID	<20:20>	AG Read Id, Source=H/Stencil=L
GSS_AG_ACCESS_TYPE	<19:18>	AG Access Type
GSS_AG_ACCESS_SIZE	<17:16>	AG Access Size
GSS_RESIDUE_LW0	<15:15>	Residue Longword 0 Valid
GSS_RESIDUE_LW1	<14:14>	Residue Longword 1 Valid
GSS_RESIDUE_LW2	<13:13>	Residue Longword 2 Valid
GSS_FIFO_TAIL_BITS	<12:11>	FIFO Tail Offset<3:2>
GSS_FIFO_IDU	<10:10>	FIFO In Drawing Unit State
GSS_EXECUTING_CLIP	<09:09>	Executing from a clip list
GSS_FIFO_BPT_STALL	<08:08>	Stalled because of breakpoint
GSS_FIFO_WFSYNC_STALL	<07:07>	Stalled because WFSYNC
GSS_FIFO_AGBUSY_STALL	<06:06>	Stalled because AG is busy
GSS_UNUSED0	<05:05>	Unused Bit
GSS_ADRS_BPT_VIRTUAL	<04:04>	Breakpoint Address Is Virtual
GSS_VM	<03:03>	Virtual Translation Go
GSS_FIFO_IDLE	<02:02>	FIFO is completely idle
GSS_AG_ACCESS	<01:01>	AG Access Breakpoint Go
GSS_FIFO	<00:00>	FIFO/Halt Go

11.5.26.1 Address Generator Busy - GSS_AG_BUSY

This bit indicates whether the Address Generator is busy (when 1) or idle (when 0). If the Address Generator has been halted from an interrupt or an exception, but still has work to perform from the current command, this bit will be asserted. This bit is asserted as soon as the Address Generator recognizes a command packet which considered an action packet. An action packet typically causes an accesses to memory (either main memory or the frame buffer). This bit is cleared when the last memory access of an action command packet is queued.

11.5.26.2 Master Graphics Interrupt - GSS_MASTER_INT

This bit indicates the state of the interrupt signal from the LCG to the CPU's interrupt controller.

11.5.26.3 In Short Circuit Mode - GSS_SHORT_CIRCUIT

This bit indicates that the command FIFO is in short circuit mode. In short circuit mode command data written to the FIFO window are sent directly to the data bus and the FIFO control state machine's residue buffer without being written to main memory.

11.5.26.4 Valid Command Packet - GSS_VALID_PACKET

This bit indicates that the current command packet data is valid and should not be ignored. This is sent to the Address Generator to signal that there is valid command data on the pixel data bus.

11.5.26.5 1st Longword Of Packet - GSS_1ST_LONGWORD

This bit indicates that the current longword is the 1st longword of the current command packet. The first longword of a command packet is particularly interesting because it contains the opcode of the command.

11.5.26.6 Command FIFO Arbitrate - GSS_FIFO_ARBITRATE

This bit reflects the state of the FIFO arbitration signal.

11.5.26.7 Command FIFO Commander Id - GSS_FIFO_COMMANDER

This field indicates from what source the FIFO control state machine is currently getting command data. Possible values of this field are:

Table 25: Possible Values of GSS_FIFO_COMMANDER

Name	Value	Description
CF_CMDR_FIFO	00	FIFO Memory
CF_CMDR_CLIP	01	Clip List Memory
CF_CMDR_NOP	10	No FIFO Commander
CF_CMDR_RESIDUE	11	FIFO Residue Buffer

11.5.26.8 AG Backwards Access - GSS_AG_BACKWARDS

This bit indicates that the Address Generator is performing a backwards access. Backwards accesses are generated when performing ROP commands with the X_BACKWARD flag set.

11.5.26.9 AG Virtual Access - GSS_AG_VIRTUAL

This bit indicates that the Address Generator is performing a virtual memory access. This signal enables the virtual translation logic. Virtual memory accesses are generated when performing ROP commands in which one of the memory setups had the VIRTUAL flag set, or when performing one of the following commands with the VIRTUAL flag set: ROP_GLYPH, FETCH_GDB, STORE_GDB, STORE_LONG.

11.5.26.10 AG Arbitrate - GSS_AG_ARBITRATE

This bit reflects the state of the Address Generator arbitration signal.

11.5.26.11 AG Read Id - GSS_AG_READ_ID

This bit indicates the operand for which the Address Generator is currently performing a memory read access. When set the current access is a source operand read, when clear the current access is a stencil operand read.

11.5.26.12 AG Access Type - GSS_AG_ACCESS_TYPE

This field indicates the type of memory access the Address Generator is currently requesting. Possible values of this field are:

Table 26: Possible Values of GSS_AG_ACCESS_TYPE

Name	Value	Description
AG_WRITE	00	Write
AG_READ	11	Read
AG_MODIFY	10	Read-Mod-Write
AG_NOP	11	No Access

11.5.26.13 AG Access Size - GSS_AG_ACCESS_SIZE

This field indicates the size of memory access the Address Generator is currently requesting. Possible values of this field are:

Table 27: Possible Values of GSS_AG_ACCESS_SIZE

Name	Value	Description
AG_LONG	00	Longword Access
AG_QUAD	01	Quadword Access
AG_OCTA	10	Octaword Access
AG_NOP	11	No Access

11.5.26.14 Residue Longword 0 Valid - GSS_RESIDUE_LW0

This bit indicates that the 1st longword of the FIFO control state machine's residue buffer contains valid command data.

11.5.26.15 Residue Longword 1 Valid - GSS_RESIDUE_LW1

This bit indicates that the 2nd longword of the FIFO control state machine's residue buffer contains valid command data.

11.5.26.16 Residue Longword 2 Valid - GSS_RESIDUE_LW2

This bit indicates that the 3rd longword of the FIFO control state machine's residue buffer contains valid command data.

11.5.26.17 FIFO Put<3:2> - GSS_FIFO_TAIL_BITS

This field contains bits <3:2> of the FIFO tail pointer FIFO_TAIL_OFFSET. These bits are used to keep track of command longwords at the tail end of the FIFO which are not octaword aligned. The FIFO get pointer FIFO_HEAD_OFFSET is always octaword aligned because the FIFO control state machine only performs octaword reads on the FIFO.

11.5.26.18 FIFO In Drawing Unit State - GSS_FIFO_IDU

This bit reflects the FIFO control state machine's in-drawing-unit state. The in-drawing-unit state indicates that the FIFO control state machine is currently executing commands from within a drawing unit. A drawing unit is a group of commands in the command FIFO which are executed once for each clipping unit (clip rectangle) in the clip list. When executing from a drawing unit the FIFO_SAVE_HEAD_OFFSET latch contains the value to load into the FIFO_HEAD_OFFSET latch to reexecute the commands in the drawing unit.

11.5.26.19 Executing From FIFO - GSS_EXECUTING_FIFO

This bit indicates whether the FIFO control state machine is executing commands from the command FIFO (when 1) or from the clip list (when 0). When the FIFO control state machine is executing commands in short circuit mode this bit is asserted.

11.5.26.20 Breakpoint Address Is Virtual - GSS_ADRS_BPT_VIRTUAL

This bit indicates that the breakpoint address contained in the BREAKPT_ADDRESS latch is a virtual address.

11.5.26.21 Virtual Translation Go - GSS_VM

This bit indicates the current state of the virtual memory translation go bit. This bit is set by writing the GO_VM bit in the LCG_GO register.

11.5.26.22 AG State Breakpoint Go - GSS_AG_STATE

This bit indicates the current state of the Address Generator state go bit. This bit is set by writing the GO_AG_STATE bit in the LCG_GO register.

11.5.26.23 AG Access Breakpoint Go - GSS_AG_ACCESS

This bit indicates the current state of the Address Generator access go bit. This bit is set by writing the GO_AG_ACCESS bit in the LCG_GO register.

11.5.26.24 FIFO Go - GSS_FIFO

This bit indicates the current state of the FIFO control state machine go bit. This bit is set by writing the GO_FIFO bit in the LCG_GO register.

11.5.27 Graphics Control Register - GRAPHICS_CONTROL

This register contains control bits for the graphics subsystem.

Table 28: Format of GRAPHICS_CONTROL

Name	Field	Description
CTRL_RESET	<31:31>	Graphics Subsystem Reset
CTRL_AG	<30:30>	AG Enable
CTRL_CLIP_LIST	<29:29>	Clip List Enable
CTRL_FIFO	<28:28>	Command FIFO Enable
CTRL_SHORT_CIRCUIT	<27:27>	Short Circuit Enable
CTRL_VM	<26:26>	Virtual Translation Enable
CTRL_VM_PROTECTION	<25:25>	VM Access Checks Enable
CTRL_OPT_INTERFACE	<24:24>	Option interface enable
CTRL_OPT_TIMOUT_SEL	<23:22>	Option timeout select
CTRL_OPT_RESET	<21:21>	Reset Option between 1 to 2 Scan Lines
CTRL_OPTION_NORESET	<20:20>	Do not reset option with graphics reset
CTRL_UNUSED1	<19:12>	Unused Bits
CTRL_AG_ACCESS_BPT_ARM	<11:11>	AG Access Breakpoint Arm
CTRL_UNUSED1	<10:10>	Unused Bit
CTRL_PACKET_BPT_ARM	<09:09>	Command Packet Breakpoint Arm
CTRL_ADDRESS_BPT_ARM	<08:08>	Address Breakpoint Arm
CTRL_UNUSED2	<07:00>	Unused Bits

11.5.27.1 Graphics Subsystem Reset - CTRL_RESET

When asserted, this bit halts the graphics system and resets it to its initialization state. The grpahics subsystem includes: the Address Generator, the command FIFO, and the clip list, but not the video subsystem. Resetting the graphics subsystem does not change the contents of the GRAPHICS_CONTROL register.

11.5.27.2 Address Generator Enable - CTRL_AG

This bit enables the Address Generator state machine. If this bit is cleared, the address generator is prevented from initiating any physical or virtural memory accesses.

11.5.27.3 Clip List Enable - CTRL_CLIP_LIST

This bit enables the FIFO control state machine to execute commands from the clip list. When this bit is clear the FIFO control state machine ignores the clip control codes in the command packets.

11.5.27.4 Command FIFO Enable - CTRL_FIFO

This enables the command FIFO. When this bit is clear the FIFO control state machine is disabled.

11.5.27.5 Short Circuit Enable - CTRL_SHORT_CIRCUIT

This bit enables the FIFO control state machine short circuit logic. In short circuit mode command data written to the FIFO window is sent directly to the data bus and the FIFO control state machine's residue buffer without being written to main memory.

11.5.27.6 Virtual Address Translation Enable - CTRL_VM

This bit enables virtual to physical address translation.

11.5.27.7 VM Page Protection Access Checks Enable - CTRL_VM_PROTECTION

This bit enables access checks against the page protection field of the PTE prior to any memory access to virtual address space. This bit should always be asserted while VMS is booted to avoid data corruption due to an error in the LCG server.

11.5.27.8 Address Generator State ARM - CTRL_AG_STATE_BPT_ARM

This bit enables the Address Generator state breakpoint. When enabled an Address Generator state breakpoint occurs when the Address Generator's current state matches the value contained in the BREAKPT_AG_STATE latch. See the description of the INT_AG_STATE_BPT bit in the GRAPHICS_INT_STATUS register for more details.

11.5.27.9 Address Generator Access ARM - CTRL_AG_ACCESS_BPT_ARM

This bit enables the Address Generator access single step breakpoint. When enabled an Address Generator access single step breakpoint occurs when the Address Generator attempts a memory access. See the description of the INT_AG_ACCESS_BPT bit in the GRAPHICS_INT_STATUS register for more details.

11.5.27.10 Command Packet Single Step ARM - CTRL_PACKET_BPT_ARM

This bit enables the command packet single step breakpoint. When enabled a command packet single step breakpoint occurs whenever a command packet is unloaded from the command FIFO or clip list. See the description of the INT_PACKET_BPT bit in the GRAPHICS_INT_STATUS register for more details.

11.5.27.11 Address Breakpoint ARM - CTRL_ADDRESS_BPT_ARM

This bit enables the address breakpoint. When enabled an address breakpoint occurs when the Address Generator attempts to access an address whose bits <29:09> match the value contained in the BREAKPT_ADDRESS latch. See the description of the INT_ADDRESS_BPT bit in the GRAPHICS_INT_STATUS register for more details.

11.5.28 Breakpoint Address Latch - BREAKPT_ADDRESS

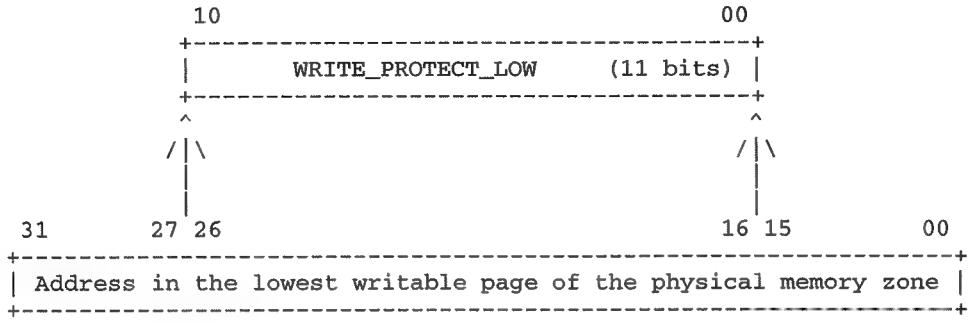
This latch is used to hold the match address for the address breakpoint. For a virtual address breakpoint bits <29:04> are significant, for a physical address breakpoint bits <27:04> are significant. When enabled, an address breakpoint occurs when the Address Generator attempts to access an address whose bits <29:04> match the contents of this latch. The BPT_ADRS_VP bit controls whether a physical (when 1) or virtual (when 0) address comparison is made.

Table 29: Format of BREAKPT_ADDRESS

Name	Field	Description
BPT_ADRS_UNUSED0	<31:30>	Unused Bits
BPT_ADRS_ADDRESS	<29:04>	Breakpoint Address
BPT_ADRS_UNUSED1	<03:01>	Unused Bits
BPT_ADRS_MATCH_L	<01:01>	Address Match Bit Low assertion
BPT_ADRS_VP	<00:00>	Virtual/Physical Select

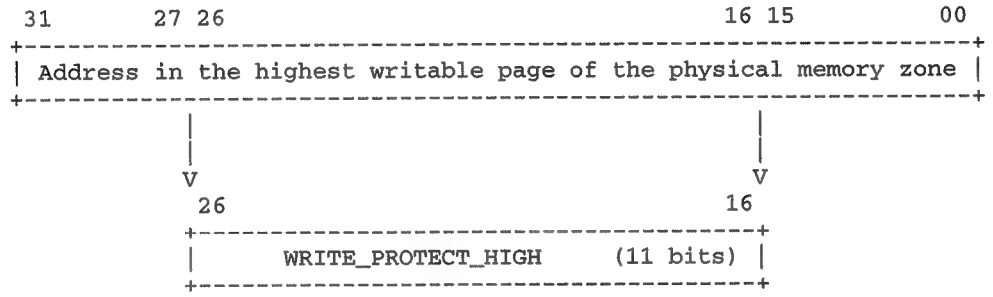
11.5.29 Write Protection Low Latch - WRITE_PROTECT_LOW

This latch contains bits <26:16> of an address in the lowest page of physical memory which can be written by the Address Generator. Because only bits <26:11> are checked by the physical memory protection logic the starting and ending pages of the allowable physical memory zone must be 64k byte aligned. The Address Generator must be stopped before writing to this register. This register can not be written to from the Command FIFO or Clip List. Note that the WRITE_PROTECT_LOW and WRITE_PROTECT_HIGH latches share the same longword address.



11.5.30 Write Protection High Latch - WRITE_PROT_HIGH

This latch contains bits <26:16> of an address in the highest page of physical memory which can be written by the Address Generator. Because only bits <26:11> are checked by the physical memory protection logic the starting and ending pages of the allowable physical memory zone must be 64k byte aligned. The LCG *must* be stopped before writing to this register. This register can not be written to from the Command FIFO or Clip List.



11.5.31 Virtual Translation Status Latch - TB_STATUS

This latch contains the status of the virtual to physical address translation logic. All bits are cleared on system reset as well as by asserting the CTRL_RESET bit in the GRAPHICS_CONTROL register.

Table 30: Format of TB_STATUS

Name	Field	Description
TBS_VM_FAULT	<31:31>	Virtual Fault
TBS_UNUSED0	<30:25>	Unused Bits
TBS_DEST_PTE_PFN	<24:24>	TB_DEST_PTE_PFN Valid
TBS_SOURCE_PTE_PFN	<23:23>	TB_SOURCE_PTE_PFN Valid
TBS_STENCIL_PTE_PFN	<22:22>	TB_STENCIL_PTE_PFN Valid
TBS_DEST_DATA_PFN	<21:21>	TB_DEST_DATA_PFN Valid
TBS_SOURCE_DATA_PFN	<20:20>	TB_SOURCE_DATA_PFN Valid
TBS_STENCIL_DATA_PFN	<19:19>	TB_STENCIL_DATA_PFN Valid
TBS_UNUSED1	<18:16>	Unused Bits
TBS_VIRTUAL_ACCESS	<15:15>	Virtual Access In Progress
TBS_VIRTUAL_LENGTH_BIT	<14:14>	Virtual Length Error Bit
TBS_ACCESS_VIOL_BIT	<13:13>	Access Violation Bit
TBS_MODIFY_BIT	<12:12>	Modify Bit
TBS_GRRAPHICS_ERROR	<11:11>	Fatal Graphics Error
TBS_DEST_ACCESS	<10:10>	Virtual Destination Access
TBS_SOURCE_ACCESS	<09:09>	Virtual Source Access
TBS_STENCIL_ACCESS	<08:08>	Virtual Stencil Access
TBS_ACCESS_MISS	<07:07>	Page Not Valid or Modify Bit Not Set or Access Violation
TBS_PTE_PTP_READ	<06:06>	PTE PFN Read In Progress
TBS_DATA_PFN_READ	<05:05>	Data PFN Read In Progress
TBS_PTE_FAULT	<04:04>	PTP Read Fault
TBS_PFN_FAULT	<03:03>	PFN Read Fault
TBS_TABLE_LENGTH_ERROR	<02:02>	Page Table Length Error
TBS_ACCESS_VIOLATION	<01:01>	Access Violation Fault
TBS_MODIFY_NOT_SET	<00:00>	Modify Bit Not Set Fault

11.5.31.1 Virtual Fault - TBS_VM_FAULT

This bit indicates that a virtual address translation fault has occured. A virtual address translation fault includes: page not valid, access violation, page table length error, and modify bit not set. This bit is equivelent to the INT_VM_FAULT bit in the GRAPHICS_INT_STATUS latch.

11.5.31.2 TB_DEST_PTE_PFN Valid - TBS_DEST_PTE_PFN

This bit indicates that the TB_DEST_PTE_PFN latch contains a valid page frame number.

11.5.31.3 TB_SOURCE_PTE_PFN Valid - TBS_SOURCE_PTE_PFN

This bit indicates that the TB_SOURCE_PTE_PFN latch contains a valid page frame number.

11.5.31.4 TB_STENCIL_PTE_PFN Valid - TBS_STENCIL_PTE_PFN

This bit indicates that the TB_STENCIL_PTE_PFN latch contains a valid page frame number.

11.5.31.5 TB_DEST_DATA_PFN Valid - TBS_DEST_DATA_PFN

This bit indicates that the TB_DEST_DATA_PFN latch contains a valid page frame number.

11.5.31.6 TB_SOURCE_DATA_PFN Valid - TBS_SOURCE_DATA_PFN

This bit indicates that the TB_SOURCE_DATA_PFN latch contains a valid page frame number.

11.5.31.7 TB_STENCIL_DATA_PFN Valid - TBS_STENCIL_DATA_PFN

This bit indicates that the TB_STENCIL_DATA_PFN latch contains a valid page frame number.

11.5.31.8 Virtual Access In Progress - TBS_VIRTUAL_ACCESS

This bit indicates that a virtual access is in progress. This bit is asserted whenever the Address Generator attempts a virtual access and remains asserted until the address translation is complete.

11.5.31.9 Length or Access Violation Error - TBS_ERROR

This bit indicates that a virtual address translation fault has occured because of an access violation or a page table length error.

11.5.31.10 Virtual Destination Access - TBS_DEST_ACCESS

This bit indicates that the current virtual address translation was for a destination operand access.

11.5.31.11 Virtual Source Access - TBS_SOURCE_ACCESS

This bit indicates that the current virtual address translation was for a source operand access.

11.5.31.12 Virtual Stencil Access - TBS_STENCIL_ACCESS

This bit indicates that the current virtual address translation was for a stencil operand access.

11.5.31.13 Virtual Access Miss - TBS_ACCESS_MISS

This bit indicates that a virtual address translation fault has occured because the data page was not valid or the modify bit of the PTE was not set.

11.5.31.14 PTP Read In Progress - TBS_PTE_PFN_READ

This bit indicates that the PFN of the process PTE which maps the adddress contained in NEXT_ADDRESS is being read from the system page table. This is a physical memory read using the address contained in PA_SPTE_PTE.

11.5.31.15 PFN Read In Progress - TBS_DATA_PFN_READ

This bit indicates that the PFN of the operand data page is being read from the process page table. This is a physical read using an address calculated from the PTE PFN in the translation buffer for the current operand and the address contained in the NEXT_ADDRESS latch.

11.5.31.16 PTP Read Fault - TBS_PTP_FAULT

This bit indicates that either a page not valid fault of a page table length error has occured.

11.5.31.17 PFN Read Fault - TBS_PFN_FAULT

This bit indicates that either a page not valid fault of a page table length error has occured.

11.5.31.18 Page Table Length Error - TBS_TABLE_LENGTH_ERROR

This bit indicates that an attempt was made to translate a virtual address which was beyond the virtual address space as specified by the MAX_VIRTUAL_ADDRESS latch.

11.5.31.19 Access PFN Violation Error - TBS_ACCESS_VIOLATION

This bit indicates that an attempt was made to access a virtual address for which the process does not have access to as specified by the protection bits in the process PTE which maps that address.

11.5.31.20 Modify Bit Not Set - TBS_MODIFY_NOT_SET

This bit indicates that an attempt was made to access a virtual address for writing and the MODIFY bit was not set in the PTE.

11.5.32 Translation Buffer Invalidate Single - TB_INVALIDATE_SINGLE

Writing a system address to this register will cause the LCG virtual memory subsystem to invalidate any translation buffer entries which have cached the translation for that address. This is a write only register.

11.5.33 Translation Buffer Invalidate All - TB_INVALIDATE_ALL

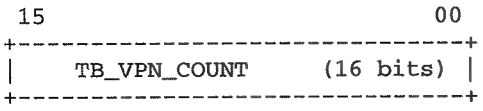
Writing any value to this bit will cause the LCG virtual memory subsystem to invalidate all of its translation buffer entries. This is a write only register.

11.5.34 Translation Buffer Invalidate Status - TB_INVALIDATE_STATUS

Whenever an the TB_INVALIDATE_SINGLE or TB_INVALIDATE_ALL registers are written to, the TB_INVALIDATE_STATUS bit is be forced to zero until the appropriate translation buffer entries have been flushed. Once the appropriate translation buffer entries have been flushed the hardware asserts this status bit. This is a read only register.

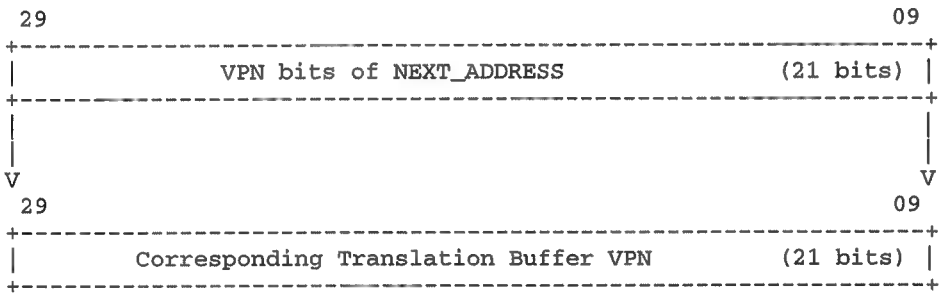
11.5.35 Graphics Translation Buffer VPN Count - TB_VPN_COUNT

This location contains the number of translation buffer entries for virtual page numbers (VPN's). Pass 1 of the S-Chip ASIC will have 6 PFNs and 3 VPNs hardwired within the chip. This location is read only.



11.5.36 VPN Latches - TB_DEST_VPN, TB_SOURCE_VPN, and TB_STENCIL_VPN

These three latches contain the virtual page number (VPN) portion of the translation buffers. The LCG has one translation buffer for each of the three possible ROP operands, destination, source, and stencil. Virtual addresses generated by the Address Generator are 28 bits wide, the VPN is taken from bits <29:09> of the virtual address and compared with the same bits from the corresponding VPN latch.



11.5.37 Next Address Latch - NEXT_ADDRESS

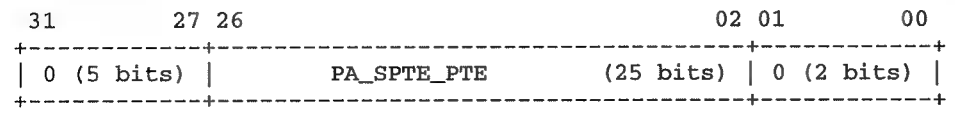
This latch contains the next address to be accessed by the Address Generator. Depending on the type of operand being accessed this address may be either physical or virtual.

11.5.38 Address of System PTE Which Maps the PTE - PA_SPTE_PTE

This latch contains the physical address of the system PTE which maps the process page table page which contains the PTE which maps the current value of NEXT_ADDRESS. This is calculated for every memory access in case it is needed. This address is used to load the appropriate PTE PFN in the translation buffer whenever it is no longer valid. The calculation which is performed is:

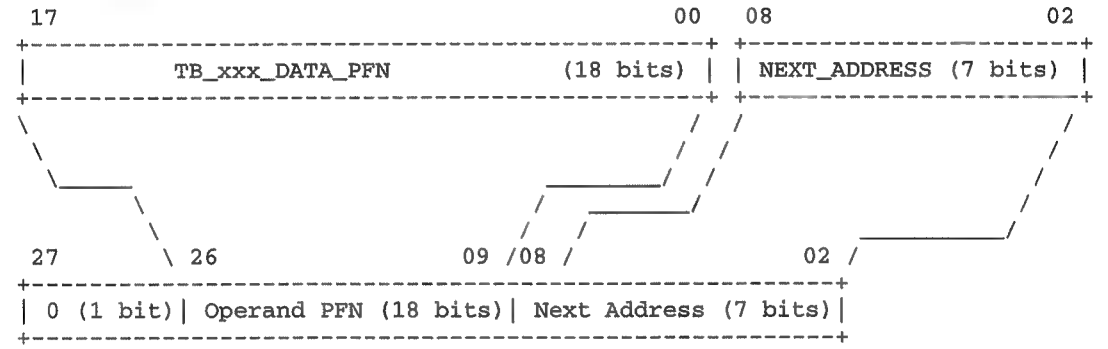
```
PA_SPTE_PTE = (PA_SPTE_P0BR<26:02> + (NEXT_ADDRESS<29:16> SHIFT_RIGHT 14))
```

Bits <29:16> of NEXT_ADDRESS are shifted right 14 bits to align with bits Literal>(<15:02>) of PA_SPTE_P0BR. The two quantities are then added together.



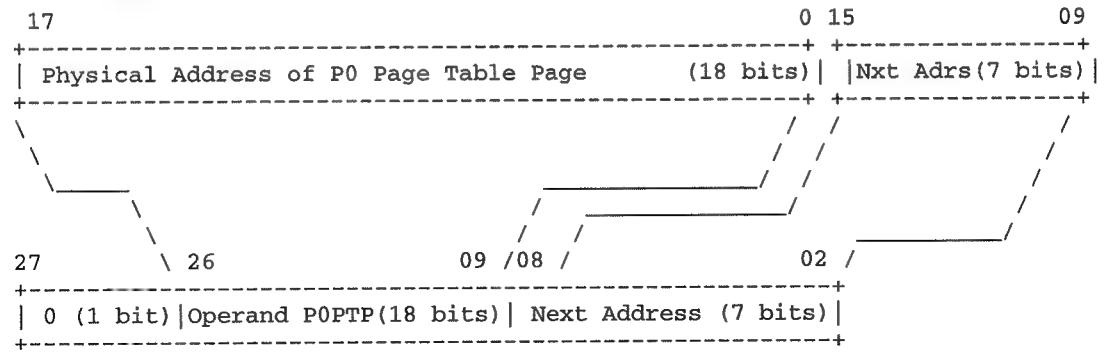
11.5.39 Data PFN Latches - TB_DEST_DATA_PFN, TB_SOURCE_DATA_PFN, and TB_STENCIL_DATA_PFN

These three latches contain the operand data page frame number (PFN) portion of the translation buffers. The LCG has one translation buffer for each of the three possible ROP operands, destination, source, and stencil. These latches are 18 bits wide, this allows for 27 bits to address 128M bytes of memory minus the lower 9 bits which specify the byte offset within a page.



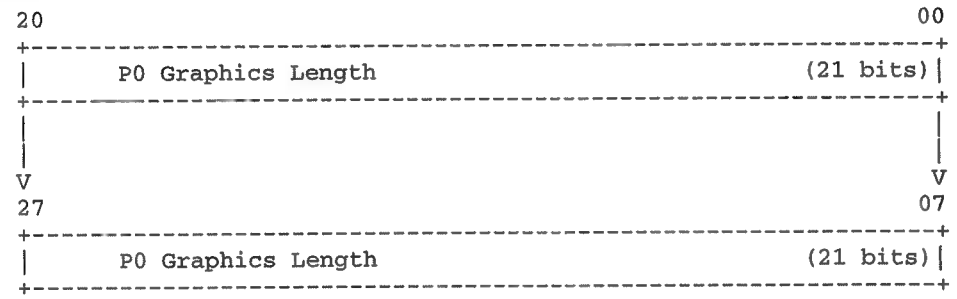
11.5.40 PTE PFN Latches - TB_DEST_PTE_PFN, TB_SOURCE_PTE_PFN, and TB_STENCIL_PTE_PFN

These three latches contain the process pate table entry (PTE) page frame number (PFN) portion of the translation buffers. The LCG has one translation buffer for each of the three possible ROP operands, destination, source, and stencil. The PFN of the process PTE is used to reload the data PFN whenever it is no longer valid. The PFN of the process PTE is obtained by looking in the system page table which is pointed to by PA_SPTE_P0BR and adding the correct offset extracted from the virtual address. These latches are 18 bits wide, this allows for 27 bits to address 128M bytes of memory minus the lower 9 bits which specify the byte offset within a page.



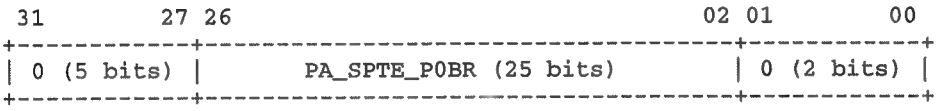
11.5.41 Max Virtual Address - MAX_VIRTUAL_ADDRESS

This latch contains the maximum virtual address of the process. This address is used to guard against attempts to access memory not currently mapped in the process page table. Before a virtual address is translated, bits <29:09> of the virtual address are compared with the corresponding bits of this latch. If the virtual address is greater than the contents of this latch a table length error fault is generated.



11.5.42 Address of System PTE Which Maps P0BR - PA_SPTE_P0BR

This latch contains the physical address of the system PTE which maps the first page of the process page table. This address is used to access PTEs within the system page table to load the PTE PFN latches in the translation buffers. To obtain the address of the system PTE which maps a desired process page table PTE the contents of this latch are added to the contents of bits <29:15> of the NEXT_ADDRESS latch. This latch is 25 bits wide which allows for 27 bits to address 128M bytes of memory minus 2 bits because the PTEs are longword aligned.



11.5.43 Command FIFO Window - Not a register or latch

Writing to this I/O space address range (2018.0000 to 201F.FFFF) uses the memory write buffering and writes data to the tail of the command FIFO. Writing to this address range while the FIFO control state machine is in short circuit mode will cause the command data to be written directly to the Address Generator and FIFO control residue buffers. Full longwords must be written to this address range. Reading any address in this range by the CPU, NI, or SCSI is unpredictable.

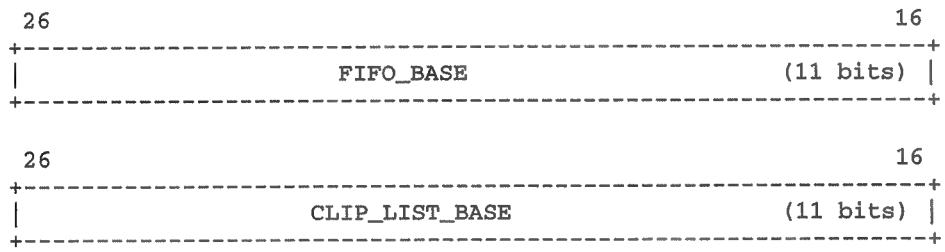
When writing command packets to this range care must be taken to pad the command longwords to an octaword boundary before wrapping to the beginning of the window. Failure to pad to an octaword will cause data corruption.

Refer to the S-Chip Specification for the details on the write buffering mechanism. Refer to the LCG Command Packet Specification for more details on the operation of the command FIFO.

11.5.44 Command FIFO and Clip List Base Latches - FIFO_BASE, and CLIP_LIST_BASE

These latches contain the base address for the command FIFO and clip list. The base address latches are concatenated with the offset latches (FIFO_HEAD_OFFSET, FIFO_TAIL_OFFSET, and CLIP_LIST_OFFSET) to address the command FIFO and clip list. The command FIFO and clip list may be located anywhere in physical main memory, but, because the base address latches contain bits <26:16> of the they may not exceed 64K bytes and must be naturally aligned. The base address latches are 11 bits wide which allows for 27 bits to address 128M bytes of memory.

Because only the offsets are incremented and not the base address latches the command FIFO and clip list may not cross a 64K boundary. Neither command FIFO nor clip list can be stored in the video frame buffer. This is because the FIFO control state machine uses octaword accesses to read command data from the command FIFO and clip list and this cannot be done with the monochrome frame buffer.

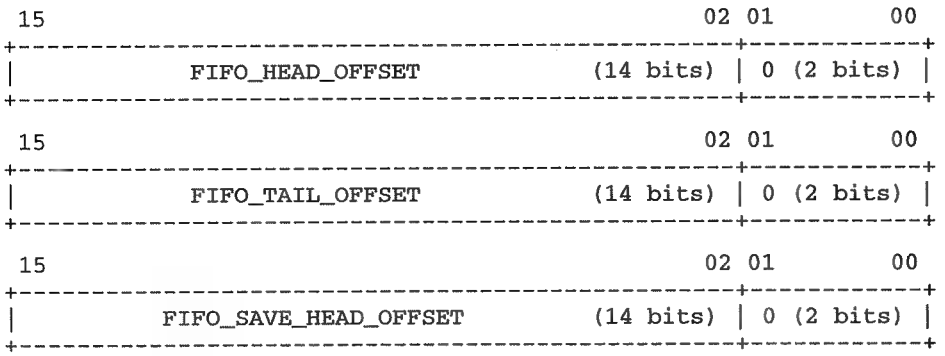


11.5.45 Command FIFO Offset Latches - FIFO_HEAD_OFFSET, FIFO_TAIL_OFFSET, and FIFO_SAVE_HEAD_OFFSET

The command FIFO head and the tail offset latches keep track of the longword offset of the head and tail of the command FIFO. The FIFO is stored as a simple ring buffer in physical main memory. Command data being inserted into the FIFO is put at the FIFO_TAIL_OFFSET. Command data is removed from the FIFO by the FIFO control state machine at the FIFO_HEAD_OFFSET. The FIFO save head offset latch keeps track of the first longword of the current drawing unit when executing a group of commands in a loop for each of the clipping units in the clip list. The FIFO_SAVE_HEAD_OFFSET is loaded into the FIFO_HEAD_OFFSET to loop back to the first command of the drawing unit when moving onto the next clipping unit in the clip list. The FIFO offset latches are 14 bits wide which allows for 16 bit offsets which are longword aligned.

The FIFO control state machine increments the contents of the offset latches as longwords are inserted/removed from the FIFO. The point at which the FIFO is wrapped is controlled by the contents of the FIFO_LENGTH_MASK field in the FIFO_MASKS latch.

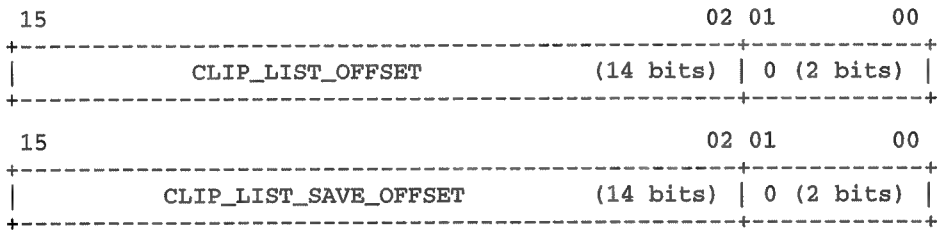
Performing a longword read on any of these offset latches, FIFO_HEAD_OFFSET, FIFO_TAIL_OFFSET, or FIFO_SAVE_HEAD_OFFSET, will return the offset concatenated with the FIFO_BASE latch forming a complete address.



11.5.46 Clip List Offset Latches - CLIP_LIST_OFFSET, and CLIP_LIST_SAVE_OFFSET

The clip list offset latch keeps track of the longword offset of the next command in the clip list. The clip list is stored as a simple linear array in physical main memory. Command data is setup in the clip list memory before establishing the clip list base address. Command data is removed from the clip list by the FIFO control state machine at the CLIP_LIST_OFFSET. The clip list save offset latch keeps track of the first longword of the clip list. After a group of commands in a drawing unit have been drawn through all the clipping units in the clip list the CLIP_LIST_SAVE_OFFSET is loaded into the CLIP_LIST_OFFSET to loop back to the first clipping unit in the clip list. The clip list offset latches are 14 bits wide which allows for 16 bit offsets which are longword aligned.

Performing a longword read on either the CLIP_LIST_OFFSET latch or the CLIP_LIST_SAVE_OFFSET latch will return the offset concatenated with the CLIP_LIST_BASE latch forming a complete address.



11.5.47 Command FIFO Length, Almost Full, Empty Mask Latch - FIFO_MASKS

This latch contains a 16 bit mask which is used for limiting the size of the command FIFO and setting up the almost full and almost empty thresholds. This latch will be cleared on System Reset.

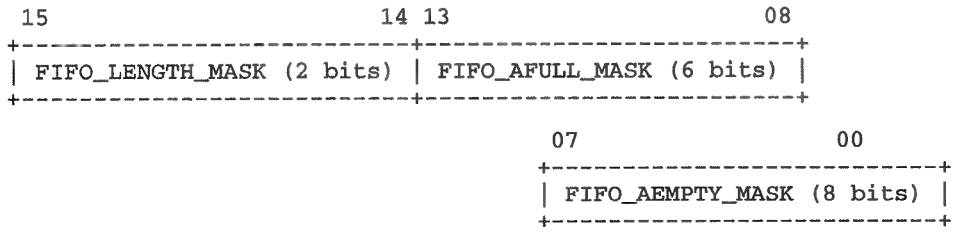


Table 31: Format of FIFO_MASKS

Name	Field	Description
FIFO_LENGTH_MASK	<15:14>	FIFO Length Mask
FIFO_AFULL_MASK	<13:08>	FIFO Almost Full Mask
FIFO_AEMPTY_MASK	<07:00>	FIFO Almost Empty Mask

11.5.47.1 FIFO Length Mask - FIFO_LENGTH_MASK

This field is used to specify the length of the command FIFO. These bits are used to mask the writing of the most significant bits when incrementing the FIFO_HEAD_OFFSET and FIFO_TAIL_OFFSET latches. Possible values of this field are:

Table 32: Possible Values of FIFO_LENGTH_MASK

Name	Value	Description
FIFO_16K	00	FIFO Length Is 16K
FIFO_32K	01	FIFO Length Is 32K
FIFO_NOT_USED	10	Not Used
FIFO_64K	11	FIFO Length Is 64K

11.5.47.2 FIFO Almost Full Mask - FIFO_AFULL_MASK

This field is used to set the FIFO almost full threshold. The FIFO_LENGTH_MASK and FIFO_AFULL_MASK bits are logically ANDed with the FIFO_LENGTH and tested for all 0s. The least significant 6 bits of the length comparison are ignored for the almost full check. When all 0s are detected the FIFO length has crossed over the almost full threshold

and the almost full interrupt (INT_FIFO_AFULL) is asserted. Possible values of this field are:

Table 33: Possible Values of FIFO_AFULL_MASK

Name	Value	Description
FIFO_AFULL_AT_64	000000	FIFO Almost Full At 64 Longwords
FIFO_AFULL_AT_128	100000	FIFO Almost Full At 128 Longwords
FIFO_AFULL_AT_256	110000	FIFO Almost Full At 256 Longwords
FIFO_AFULL_AT_512	111000	FIFO Almost Full At 512 Longwords
FIFO_AFULL_AT_1024	111100	FIFO Almost Full At 1024 Longwords
FIFO_AFULL_AT_2048	111110	FIFO Almost Full At 2048 Longwords
FIFO_AFULL_AT_4096	1111111	FIFO Almost Full At 4096 Longwords

11.5.47.3 FIFO Almost Empty Mask - FIFO_AEMPTY_MASK

The output of the comparitor for the almost full check described above is inverted resulting in the number of longwords written minus 1, or approximately the number of longword written to the command FIFO. The FIFO_AEMPTY_MASK bits are logically ANDed with the inverted length and checked for all 0s. When all 0s are detected the FIFO length has crossed under the almost empty threshold and the almost empty interrupt (INT_FIFO_AEMPTY) is asserted. Possible values of this field are:

Table 34: Possible Values of FIFO_AEMPTY_MASK

Name	Value	Description
FIFO_AEMPTY_AT_32	00000000	FIFO Almost Empty At 32 Longwords
FIFO_AEMPTY_AT_64	10000000	FIFO Almost Empty At 64 Longwords
FIFO_AEMPTY_AT_128	11000000	FIFO Almost Empty At 128 Longwords
FIFO_AEMPTY_AT_256	11100000	FIFO Almost Empty At 256 Longwords
FIFO_AEMPTY_AT_512	11110000	FIFO Almost Empty At 512 Longwords
FIFO_AEMPTY_AT_1024	11111000	FIFO Almost Empty At 1024 Longwords
FIFO_AEMPTY_AT_2048	11111100	FIFO Almost Empty At 2048 Longwords
FIFO_AEMPTY_AT_4096	11111110	FIFO Almost Empty At 4096 Longwords
FIFO_AEMPTY_AT_8192	11111111	FIFO Almost Empty At 8192 Longwords

11.5.47.4 FIFO Full/Empty Detection

The FIFO length is also checked without the mask bits, but with the FIFO_LENGTH_MASK bits enabled. When the result of the comparison is 0 and the FIFO almost full interrupt bit (INT_FIFO_AFULL) is set, the FIFO is full and the FIFO full interrupt (INT_FIFO_FULL) will be asserted.

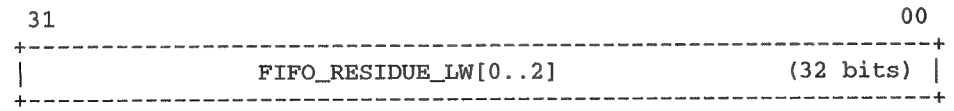
When the unmasked comparison is 0 and the FIFO almost full interrupt bit (INT_FIFO_AFULL) is clear, the FIFO is empty and the FIFO empty interrupt (INT_FIFO_EMPTY) is asserted. Note that there still may be some longwords (less than an octaword) of command data still in the FIFO. The field GSS_FIFO_TAIL_BITS in the

GRAPHICS_SUB_STATUS register can be used to determine if there are any longwords remaining in the FIFO.

11.5.48 Command FIFO Residue Latches - FIFO_RESIDUE_LW[0..2]

These three longword latches make up the FIFO control state machines residue buffer. When the FIFO control state machine reads an octaword from the command FIFO it processes the first longword "in hand" and temporarily stores the other three longwords in its residue buffer.

When the FIFO control state machine is operating in short circuit mode writes to the FIFO window are directed to the residue buffer. When leaving short circuit mode, such as when the clip list is enabled, any valid command data which remains in the residue buffer is discarded.



11.5.49 Command FIFO Length - FIFO_LENGTH

This location contains the output of the FIFO_HEAD_OFFSET and FIFO_TAIL_OFFSET comparator. This 14 bit quantity represents the number of longwords available in the command FIFO. The comparator performs the following operation:

```
FIFO_LENGTH= (FIFO_HEAD_OFFSET - FIFO_TAIL_OFFSET)
```

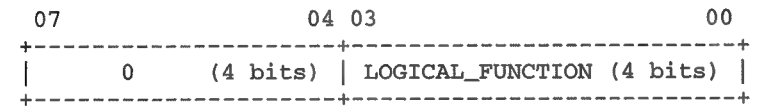
except when the GSS_FIFO_IDU bit in the GRAPHICS_SUB_STATUS register is set. Which means that the FIFO_SAVE_HEAD_OFFSET latch contains the offset of a command to which we intend to loop back and reexecute through the next clipping unit. In this instance, the comparator uses:

```
FIFO_LENGTH= (FIFO_SAVE_HEAD_OFFSET - FIFO_TAIL_OFFSET)
```

In both cases the hardware varies the equation slightly when the tail offset is greater then the head offset (the FIFO has wrapped). The FIFO_LENGTH_MASK mask IN the FIFO_MASKS latch are used to mask the resulting value.

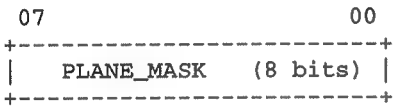
11.5.50 Graphics Pixel Function Latch - LOGICAL_FUNCTION

This 4 bit latch contains the logical function to be used by the pixel SLU. The standard X Window encoding of the 16 boolean operations is used, these are described within the pixel SLU section.



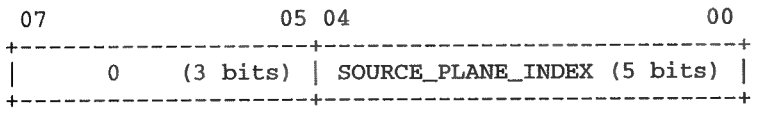
11.5.51 Graphics Plane Mask Latch - PLANE_MASK

This 8 bit latch contains the plane mask used by the pixel SLU to determine which planes to write. Plane masking will work with operands in both main memory (DRAM) and the video frame buffer (VRAM). The PLANE_MASK mask must be set to either all 1s or all 0s when writing to a monochrome operand. Plane masking for main memory (DRAM) operands will be accomplished by using read-modify-writes.



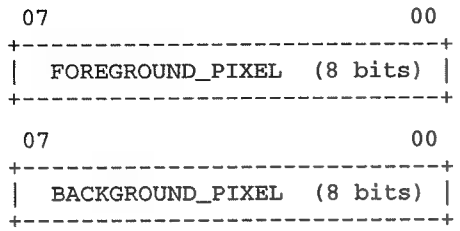
11.5.52 Source Plane Index Latch - SOURCE_PLANE_INDEX

This 5 bit latch contains the index of the bit plane within a pixel to be used from the source operand for plane extraction. LCG-I only implements bits <02:00>.



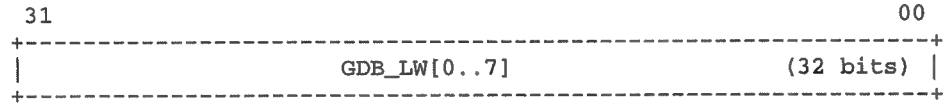
11.5.53 Graphics Foreground and Background Pixel Latches - FOREGROUND_PIXEL, and BACKGROUND_PIXEL

These two 8 bit latches contain the foreground and background pixel values. These are used as constant latches within the pixel SLU for fill and pixel expansion operations. These latches must be set to either all 1s or all 0s for monochrome destination operands.



11.5.54 Graphics Data Buffer Latches - GDB_LW[0..7]

These 8 latches make up the two octaword data buffer which can be loaded by the CPU or used as temporary storage for block moves, as in rasterops, scrolling, fill rectangles, tiling, stipples, and text. The necessity of buffering the source is to aid in shifting source data. These latches are not normally used directly by the VAX CPU and command packets.



11.5.55 Pixel SLU State Latch - SLU_STATE

This 8 bit latch contains state bits for the pixel SLU.

Table 35: Format of SLU_STATE

Name	Field	Description
SLU_UNUSED0	<15:06>	Unused bits
SLU_STENCIL_ENABLE	<05:05>	Stencil operand enabled
SLU_SOURCE_ENABLE	<04:04>	Source operand enabled
SLU_MONO	<03:03>	Mono/Color
SLU_USE_GDB	<02:02>	Use graphics data buffer
SLU_PIXEL_EXPAND	<01:01>	Pixel Expand
SLU_TRANSPARENT	<00:00>	Transparent

11.5.55.1 Stencil operand enabled - SLU_STENCIL_ENABLE

This bit when set indicates that the stencil operand is being used by the current ROP. The SLU masks writes to the destination pixels using the bits of the stencil operand.

11.5.55.2 Source operand enabled - SLU_SOURCE_ENABLE

This bit when set indicates that the source operand is being used by the current ROP. The SLU fetches source data from the graphics data buffer and shifts it to align with the destination before combining it according to the logical function in use.

11.5.55.3 Mono/Color - SLU_MONO

This bit indicates the depth of the destination operand. The destination may be monochrome (when 0) or color (when 1).

11.5.55.4 Use Graphics Data Buffer - SLU_USE_GDB

This bit when set indicates that the source for the current ROP is to be fetched from the graphics data buffer. When this bit is clear the source pixel data is taken from the FOREGROUND_PIXEL latch.

11.5.55.5 Pixel Expand - SLU_PIXEL_EXPAND

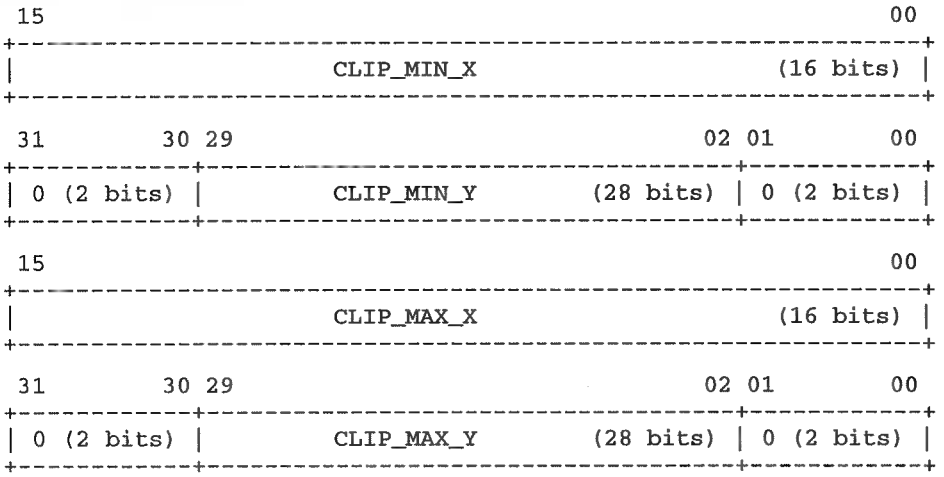
This bit when set indicates that the current ROP is to be performed using pixel expand mode. In pixel expand mode a monochrome source, such as a stipple or glyph, is expanded to color pixels by using the value of the FOREGROUND_PIXEL latch whereever the source has a pixel set and using the value of the BACKGROUND_PIXEL latch whereever the source has a pixel clear.

11.5.55.6 Transparent - SLU_TRANSPARENT

This bit when set indicates that the current ROP is to be performed using transparent mode. In transparent mode the pixels of a monochrome source are used to mask the writing of the destination pixels. Only where the source pixels are set are the destination pixels written. This mode is often used with stipples and glyphs.

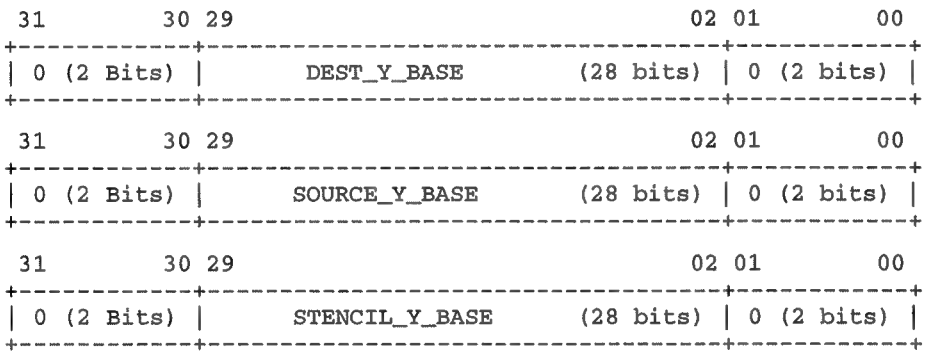
11.5.56 Clipping Rectangle Latches - CLIP_MIN_X, CLIP_MIN_Y, CLIP_MAX_X, and CLIP_MAX_Y

These four latches are used to specify the bounds of the current clip rectangle. The clip rectangle is only used for the destination operand. The X values of the clip rectangle are specified as the minimum and maximum X coordinates which are clipped in, specified in pixels. The Y values of the clip rectangle are specified as the minimum and maximum Y coordinates which are clipped in, specified as the longword address of the Y scanlines. relative to the pixel map left edge. The X latches and comparator are 16 bits wide. The Y latches and comparator are 28 bits wide.



11.5.57 Y Base Latches - DEST_Y_BASE, SOURCE_Y_BASE, and STENCIL_Y_BASE

These three latches contain the linear longword address of the beginning scanline for each of the three ROP operands. The base address latches are 28 bits wide.



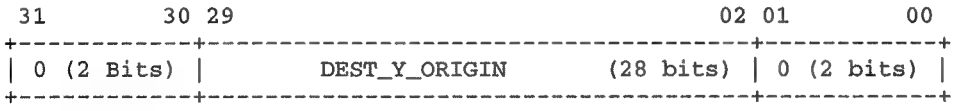
11.5.58 Destination Y Origin Latch - DEST_Y_ORIGIN

This latch is loaded from the DEST_SETUP command packet with the destination operand's Y origin. The Y origin is the address of the scanline where the Y coordinate equals zero.

The DEST_SETUP command packet contains a DEST_Y_ORIGIN and the ROP commands contain a DEST_Y_OFFSET. When processing a ROP command the following operation occurs:

```
DEST_Y_BASE = (DST_Y_ORIGIN + DEST_Y_OFFSET)
```

This mechanism is used to allow each clip rectangle specified in a clip list to potentially be stored in different sections of memory. This simplifies the implementation of things such as backing store.



11.5.59 X Position Latches - DEST_X, DEST_X0, and NEXT_X

These three latches contain different instances of the current X position represented as a pixel offset from the beginning of the current scanline. These latches are signed 16 bit quanities. The output of these latches is multiplexed, depending on whether the destination operand is monochrome or color.

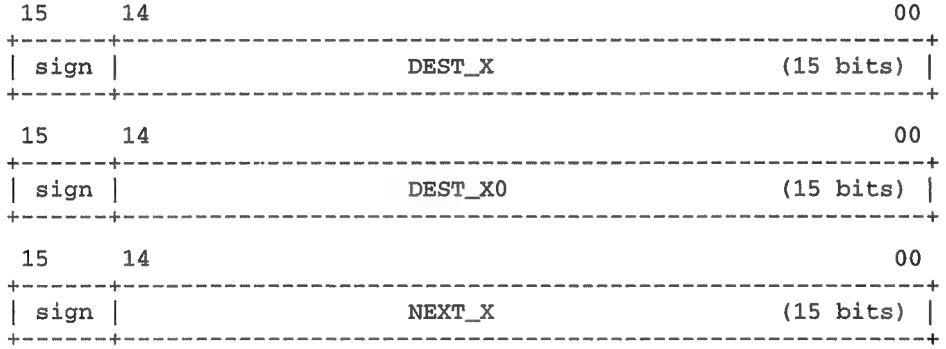
The DEST_X latch contains the current X position with respect to the destination operand.

The DEST_X0 latch contains the initial value of DEST_X for a given ROP and is loaded into the DEST_X latch to return to the starting X position.

The NEXT_X latch is used to temporarily store the sum of DEST_X and DEST_X_BIAS.

All three operands are addressed relative to the current destination X position, that is DEST_X. The actual addresses accessed are generated using the following equations:

```
DEST Linear Address    = DEST_X + DEST_X_BIAS + DEST_Y_BASE
SOURCE Linear Address  = DEST_X + SOURCE_X_BIAS + SOURCE_Y_BASE
STENCIL Linear Address = DEST_X + STENCIL_X_BIAS + STENCIL_Y_BASE
```

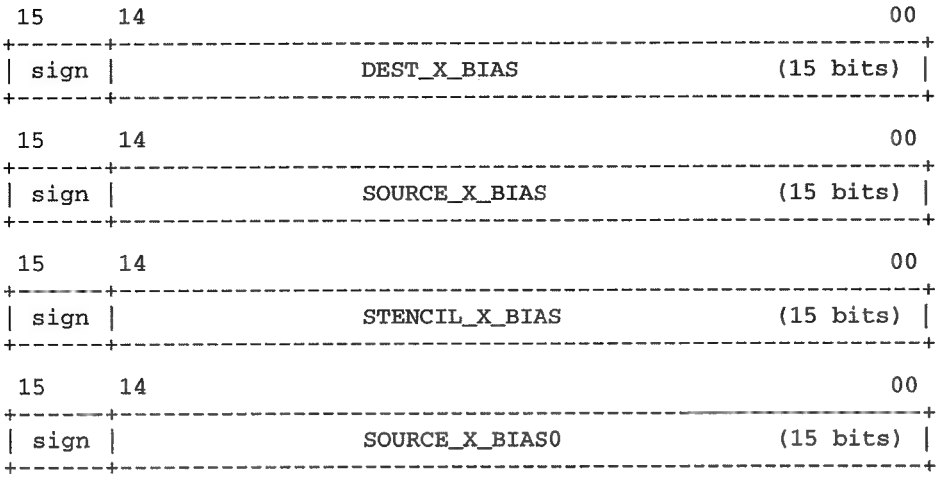


11.5.60 X Bias Latches - DEST_X_BIAS, SOURCE_X_BIAS, STENCIL_X_BIAS, and SOURCE_X_BIAS0

These three latches are used to normalize the three operands to a single X coordinate space. There is an X bias latch for each of the three possible operands. The X bias for a given operand is the delta between the current value of DEST_X and the X coordinate for that operand. Software sets up the X bias values, taking into account both memory alignment and the difference between the operand and destination X offsets.

The source operand has both a current X bias and an initial X bias. The initial X bias is loaded by the SOURCE_SETUP command packet and is used at the beginning of each scanline. The current X bias latch is used for keeping track of the X bias from the X position in the X direction. The reason for the difference from the initial X bias is to account for shifting the source data. The SOURCE_X_BIAS latch is also used for tiles and stipples to handle tile rotation.

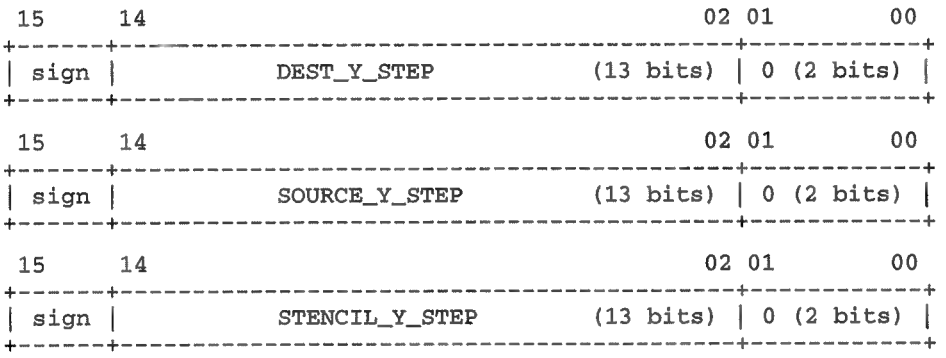
The STENCIL_X_BIAS latch is also used when drawing vectors as the error accumulation register. The most significant bit is used for determining the selection of the primary and secondary error offset latches for the next iteration. Bit 15 also directs the updating of the DEST_X and DEST_Y_BASE latches.



11.5.61 Y Step Latches - DEST_Y_STEP, SOURCE_Y_STEP, STENCIL_Y_STEP

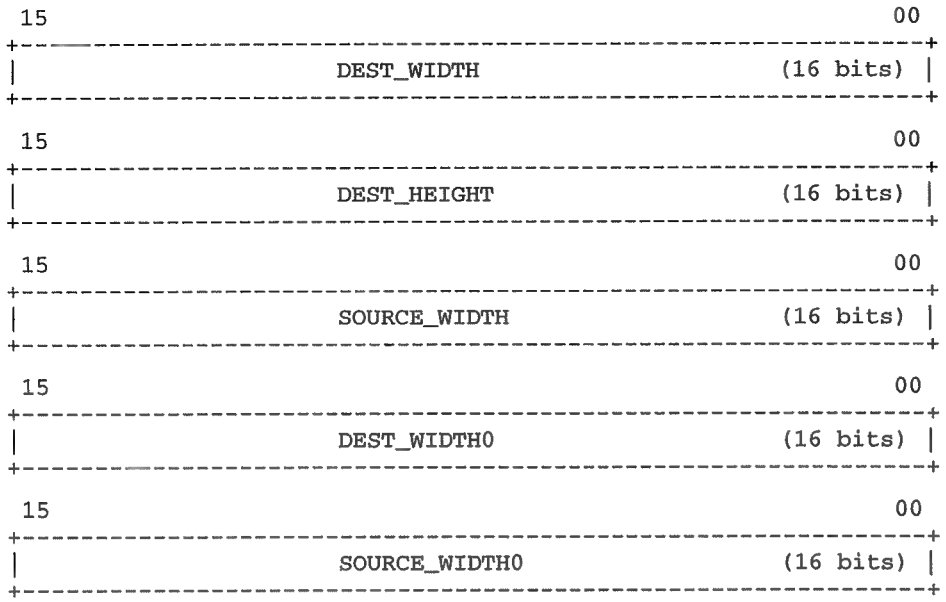
These three latches contain the Y step of the three operands. The Y step for a given operand is the number of bytes from [x,y] to [x,y+1] for that operand. Said another way it is the number of bytes in a scanline including any padding. The Y steps are stored as 16 bit signed quantities.

Both the DEST_Y_STEP and the SOURCE_Y_STEP are used in vector drawing for the positive and negative Y steps.



11.5.62 Operand Counters - DEST_WIDTH, DEST_HEIGHT, SOURCE_WIDTH, DEST_WIDTH0, and SOURCE_WIDTH0

These five latches are length counters for the source and destination operands. All counters count down and are pixel quanities. The destination operand requires a remaining width and a height, as well as, a starting width. Some sources (tiles and stipples) require a remaining width and a starting width. The DEST_WIDTH is also used for drawing vectors as the number of pixels to be drawn. These are implemented as 5 input latches, 1 decrement combinational logic, and 1 output latch for minimal gates.



11.5.63 TILE_WIDTH

TBS - Blaise

11.5.64 TILE_ROTATION

TBS - Blaise

11.5.65 Address Generator Current State Latch - AG_CURRENT_STATE

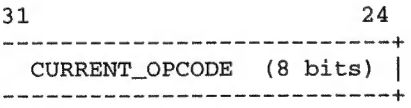
This 8 bit latch contains the Address Generator state machine's current state. This may be read for debugging purposes when using the Address Generator state breakpoint.

11.5.66 Action Code Latch - OP_ACTION_CODE

This 8 bit latch contains the operation action code. The action code specifies the basic type of operation being performed and the depths of the source and destination operands. See the discussion of the OP_SETUP command in the LCG Command Packets Specification.

11.5.67 Current Opcode Latch - CURRENT_OPCODE

This latch contains the opcode of the command packet currently being executed.



11.5.68 Operation Longwords Count Latch - OP_LW_COUNT

TBS - Blaise This 8 bit latch contains the number of longwords remaining to be processed for the current command packet.

11.5.69 Address Generator Status Latch - AG_STATUS

This 16 bit latch contains the status bits for the Address Generator state machine. This latch is read only.

Table 36: Format of AG_STATUS

Name	Field	Description
AG_UNUSED0	<31:30>	Unused Bits
AG_START	<29:29>	About to start a flow from a command packet
AG_BUSY	<28:28>	Address generator is busy drawing
AG_ADDRESS_LATCHED	<27:27>	Virtual subsystem has taken the next address
AG_FIRST_WRITE	<26:26>	First access of a scanline in destination
AG_SIGN_OF_X	<25:25>	Sign of destination X
AG_X_REVERSE	<24:24>	X operation is going in reverse direction
AG_Y_REVERSE	<23:23>	Y operation is going in reverse direction
AG_Y_NO_UPDATE	<22:22>	Do not change y after current command
AG_X_UPDATE	<21:21>	Change X after current command
AG_DONT_ARB_TEXT	<20:20>	We are doing text, do not arbitrate
AG_DONT_ARB_LINE	<19:19>	We are doing a line, do not arbitrate
AG_STENCIL_X_BIAS	<18:18>	Stencil X bias has been adjusted by access size
AG_SOURCE_X_BIAS	<17:17>	Source X bias has been adjusted by access size
AG_DEST_WIDTH_LEZ	<16:16>	Destination width is less than or equal to zero
AG_SOURCE_WIDTH_LEZ	<15:15>	Source width less than or equal to zero
AG_NEW_TILE	<14:14>	Need get first part of tile again
AG_STENCIL_COUNT	<13:13>	Pixels used by stencil mod 16 / 16
AG_X_CARRY_TERM_OUT	<12:12>	Carry out of the X adder
AG_START_MASK	<11:11>	First inked access of a Scanline uses mask
AG_REPEAT	<10:10>	Somewhere after the first access of a tile
AG_X_REJECT_0	<09:09>	X has been rejected on the near side of the copy
AG_Y_REJECT_0	<08:08>	Y has been rejected on the near side of the copy
AG_X_REJECT_1	<07:07>	X has been rejected on the far side of the copy
AG_Y_REJECT_1	<06:06>	Y has been rejected on the far side of the copy
AG_DEST_VIRTUAL	<05:05>	Destination is virtual
AG_SOURCE_VIRTUAL	<04:04>	Source Virtual
AG_STENCIL_VIRTUAL	<03:03>	Stencil Virtual
AG_AUTO_MOD	<02:02>	AG in automod mode
AG_XBS2_BUMPED	<01:01>	Stencil Xbias was bumped by 32 pixels
AG_NO_DRAW	<00:00>	Current op is no draw

11.5.69.1 Single Plane Source - AG_MONO_SOURCE

This bit indicates that the source operand is a single plane.

11.5.69.2 First Read-Mod-Write - AG_FIRST_RMW

This bit indicates that the next access will be the first read-mod-write.

11.5.69.3 Source Operand Enabled - AG_SOURCE_ENABLED

This bit indicates that the current ROP requires a source operand.

11.5.69.4 First Source Read - AG_FIRST_SOURCE_READ

This bit indicates that the next source access will be the first.

11.5.69.5 Source Loop-Back - AG_SOURCE_LOOPBACK

This bit is used for hardware debug.

11.5.69.6 Source Count - AG_SOURCE_COUNT

This bit indicates TBS - Blaise.

11.5.69.7 Shadow of MEM_DONE - AG_MEM_DONEM

This bit shadows the MEM_DONE signal for memory arbitration.

11.5.69.8 Source Read Needed - AG_NEED_SOURCE

This bit indicates that a source read is required.

11.5.69.9 Stencil Operand Enabled - AG_STENCIL_ENABLED

This bit indicates that the current ROP uses the stencil operand.

11.5.69.10 Error - AG_ERROR

This bit indicates that and Address Generator error has occured.

11.5.69.11 Scanline Start - AG_SCANLINE_START

This bit indicates that the Address Generator is at the start of a scanline.

11.5.69.12 Octant<02> - AG_OCTANT_BIT2

This bit is bit 2 of the octant specified for a vector.

11.5.69.13 Stencil Loop-Back - AG_STENCIL_LOOPBACK

This bit is used for hardware debug.

11.5.69.14 Octant<01> - AG_OCTANT_BIT1

This bit is bit 1 of the octant specified for a vector.

11.5.69.15 Stencil Count - AG_STENCIL_COUNT

This bit indicates TBS - Blaise.

11.5.69.16 Octant<00> - AG_OCTANT_BIT0

This bit is bit 0 of the octant specified for a vector.

11.5.69.17 Single Plane Destination - AG_MONO_DEST

This bit indicates that the destination operand is a single plane.

11.5.70 Operand Shift Latches - DEST_SHIFT_TERM, SOURCE_SHIFT_TERM, STENCIL_SHIFT_TERM, SOURCE_SHIFT_COUNT, and STENCIL_SHIFT_COUNT

These 16 bit latches are used to coordinate the shifting of the operand data.